COMPUTING INTERESTING TOPOLOGICAL FEATURES

BY

ERIN WOLF CHAMBERS

B.S., University of Illinois at Urbana-Champaign, 2002
M.S., University of Illinois at Urbana-Champaign, 2006

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Associate Professor Jeff Erickson, Chair
Professor Robert Ghrist
Associate Professor Steven Lavalle
Professor John Hart
Professor Nina Amenta, University of California at Davis

Abstract

Many questions about homotopy are provably hard or even unsolvable in general. However, in specific settings, it is possible to efficiently test homotopy-equivalence or compute shortest cycles with prescribed homotopy. We focus on computing such "interesting" topological features in three settings. The first two results are about cycles on surfaces; the third is about classes of homotopies in $\mathbb{R}^2$ minus a set of obstacles; and the final result is about paths and cycles in Rips complexes.

First, we examine two problems in the combinatorial surface model. Combinatorial surfaces combine properties of graphs and manifolds, making a rich set of techniques available for analysis and algorithm design. We give algorithms to find the shortest noncontractible and nonseparating cycles in a combinatorial surface in $O(g^3 n \log n)$ time. Our main tool is a data structure that kinetically maintains the shortest path tree as the root of the tree moves around the vertices of a single face. The total running time is $O(g^2 n \log n)$. By maintaining the data structure persistently, we can answer shortest path queries in $O(\log n)$ time.

Next we consider finding the shortest *splitting* cycle in a combinatorial surface, or simple cycle which is both separating and noncontractible; such cycles divide the topology of the surface as well as the underlying graph. We prove that finding the shortest splitting cycle is NP-Hard. We then give an algorithm that runs $g^{O(g)} n \log n$ time, which is polynomial if the surface is fixed.

We then examine a very different setting, namely similarity between curves in some underlying metric space. If we imagine a homotopy between the curves as a way to morph one curve into the other, we can optimize the morphing so that the maximum distance any point must travel is minimized. This is a generalization of the more well known Fréchet distance, with the additional requirement that the leash to move continuously in the ambient space. We call this distance the *homotopic Fréchet distance*. We give a polynomial time algorithm to compute the homotopic Fréchet distance between two curves in the plane minus a set of polygonal obstacles. We also extend our characterization of optimal morphings to surfaces of nonpositive curvature.

Finally, we examine a more fundamental homotopy problem in a different setting. A *Rips complex* is a simplicial complex defined by a set of points from

some metric space where every pair of points within distance 1 is connected by an edge, and every $(k+1)$-clique in that graph forms a k-simplex. We prove that the projection map which takes each k-simplex in the Rips complex to the convex hull of the original points in the plane induces an isomorphism between the fundamental groups of both spaces. Since the union of these convex hulls is a polygonal region in the plane, possibly with holes, our result implies that the fundamental group of a planar Rips complex is a free group, allowing us to design efficient algorithms to answer homotopy questions in planar Rips complexes.

For my husband, who got me through it all,
and for my parents, who got me here in the first place.

Acknowledgements

This thesis would not have happened without help from countless people. First and foremost, thanks to Jeff Erickson, who has been both advisor and mentor. Without his guidance, advice, and the occasional kick in the pants, I would not have finished. I owe much of my ability as a researcher and an educator to his example and his teaching, and cannot imagine a better guide through grad school.

The work presented in this thesis is due to collaboration with various individuals, including Sergio Cabello, Éric Colin de Verdière, Vin de Silva, Jeff Erickson, Robert Ghrist, Sylvain Lazard, Francis Lazarus, Shripad Thite, and Kim Whittlesey. My other collaborators include Douglas West, Tanya Crenshaw, Dan Cranston, Kevin Milans, David Bunde, Heather Metcalf, Umesh Thakkar, Pratik Worah, Bill Kinnersley, Noah Prince, and Piotr Adamcyzk. Working with such outstanding people has made the journey a pleasure, and I am grateful for the opportunity to work with each of them.

Thanks to my fellow students in the theory group. In particular, I would never have even made it through quals without the guys from my year: Matthew Belcher, Dan Cranston, John Fischer, and Kevin Milans. Thanks for all the years of studying and working together; wherever we wind up, I'm glad we all started together. David Bunde, Bardia Sadri, and Shripad Thite were the senior grad students who provided both examples and advice on how to make it through. Later on, Nitish Korula and Mike Rosulek were always willing to give help with teaching, writing, and LaTeX. They, along with Tracy Grauman, Feidha Zhu, Ke Chen, Pratik Worah, and Hemanta Maji, were the reason I had to avoid the office to get work done; I will miss our discussions and debates.

All of the faculty in the theory group have helped me improve numerous presentations and papers, and they have always been ready to listen to a problem or answer a question when I needed it. In particular, Edgar Ramos, Sariel Har-Peled, and Chandra Chekuri have been the extra advisors who were always available. Thanks also to Brian Bailey and David Forsythe for mock interviews and helpful advice, and to Steve Lavalle, John Hart, Rob Ghrist, and Nina Amenta for serving on my committee.

Many other friends provided advice, reviews, and sometimes a shoulder to cry on when needed. In particular, Tanya Crenshaw and Kim Belcher are the

reason this thesis was actually written. Jodie Boyer, Eric Owiesny, Shamsi Iqbal, Anna Yershova, and Jacob Biehl have also been there to commiserate, motivate, and advise. I will miss them all as we leave for greener pastures. Outside of this program, my many wonderful friends, particularly Lizzie Codie, Kavitha Sagi, Mel White, Ray Reskusich, Julie Snyder, Meghan Meharry, Darren Hron, Tim Skirvin, and Rebecca McNulty, have helped keep me grounded and sane for my entire time here.

I've also learned that no journey is possible without people to help you through the details. Barb Cicone, Mary Beth Kelly, and Kay Tomlin have all helped with both academic advising and life advice. Thanks also to the administrative assistants who talked me through all the paperwork and difficulties: Elaine Wilson, Holly Bagwell, Angie Bingham, Ellen Corcoran, and Kim Osmond. They are what keeps this department going, and they are appreciated more than they will ever know.

Thanks to the many other mentors who have helped me throughout the years. Even if they didn't realize it at the time, they are much of the reason I had the courage to take this path and see it through. In particular, I would like to thank Tanya Berger-Wolf, Klara Nahrstedt, Robin Kravets, and Nina Amenta, who have all provided words of encouragement when they were most needed.

And finally, my family has been my greatest support throughout this whole process. My husband Jeff is the most amazing husband and father, and is the reason that it all could happen. I cannot imagine going through life with a better partner. My daughter Grace provided the smile at the end of a long day which made everything better; some day, I hope she will be as proud of me as I already am of her. My guinea pig Boca was also there to help out by nibbling on any math homework that was stressing me out. My brother Erik is the one who started me in computer science years ago; thanks for the encouragement and for believing in me. Thanks to my sisters, Kristin and Kara, who put up with all the whining about how I would never be able to do this and somehow managed to make it all seem possible by the end of the conversation. Thanks also to my parents, who gave me the example and the courage to follow my dreams.

Table of Contents

Chapter 1

Introduction

1.1 Computational Topology

Computational topology is an exciting and interesting area at the intersection of mathematics and computer science. Historically, algorithmic techniques abound in topology, extending back to the algorithmic proof by Dehn and Heegaard of the surface classification theorem in 1907 [74] and the algorithm by Dehn in 1911 to test if a cycle on a surface is contractible [40]. Perhaps the first analysis of a topological algorithm was in 1961, when Haken proved that testing if a knot is trivial takes at most quadruply exponential time [71]. Until recently, however, relatively little thought was given to the efficiency of such algorithms or the applicability of the techniques.

Topological questions occur in many areas of computer science. In graphics, objects are modeled as meshes formed from scanned point set surfaces. Configuration spaces are a type of topological space used for robot motion planning. Data mining techniques rely on the assumption that the distribution of points lies on some low dimensional manifold embedded in high dimensional spaces. Computational topology thus combines tools and techniques from computer science disciplines, including computational geometry, robotics, graphics, and networking, with mathematical tools from geometry, algebraic topology, differential topology, combinatorics, and many other areas.

Many applications from computer science require information about the topological structure of an object. Before we examine topological features in a particular setting, it is necessary to discuss what type of topological space we will work in. One type of topological space which we will focus on in this thesis is a manifold. An n-dimensional manifold is a topological space where every point has a neighborhood homeomorphic to n-dimensional Euclidean space. (See Chapter 2 for precise definitions.) Surfaces, or 2-manifolds, are particularly useful for modeling 3-dimensional shapes. A fundamental result in topology, called the classification theorem, states that a surface is uniquely characterized by its genus, number of boundaries, and orientability; for a standard reference, see Stillwell [112]. For example, the disk is the only orientable genus one 2-manifold with one boundary; the sphere is the only orientable 2-manifold with genus zero and no boundary; and the Möbius strip is the only nonorientable

genus one surface with one boundary.

Surfaces can be represented in many ways. Surfaces can be represented implicitly as the zero set of a function. From a computational perspective, however, implicit surfaces are difficult to deal with, in part because geodesics do not have closed forms, so we are forced to use numerical approximation techniques [100, 30, 23]. Polyhedral surfaces are formed by gluing together simple, closed polygons along their edges. A polyhedral surface is piecewise linear if the local metric within each such polygon is Euclidean. Computationally, even computing shortest paths in a piecewise linear surface can be difficult; it is possible for a shortest path in a size $O(1)$ piecewise linear surface to intersect the edges of the 1-skeleton an unbounded number of times; see [57] for details.

Even more coarsely, surfaces can be represented by specifying a graph of the edges and vertices, along with face information, without specifying any areas or distances inside of the faces. This model is called the combinatorial surface model, and it is the basis for many of the results in this thesis. This model reduces questions about surface topology to computations in a graph. In particular, shortest paths on a combinatorial surface can be computed using Dijkstra's algorithm [46]. In addition, in many graphics algorithms that cut surfaces along paths or cycles, paths are restricted to the edges of the mesh rather than crossing faces (see e.g. [68, 108, 109, 111]), so algorithms on a combinatorial surface mirror the current techniques quite accurately.

There are many ways to transform input data into a combinatorial surface representation. In graphics, the given data is often a set of points which are sampled (possibly with noise) from some underlying object. Algorithms typically transform these points into a mesh that represents the underlying surface; we may then use the edges of the mesh as the graph for the combinatorial surface. To compute meshes, some algorithms explicitly compute a surface from the input points, often by extracting a subcomplex of the Delaunay triangulation or Voronoi diagram [51, 4]. Given appropriate sampling constraints on the points, based on local feature size of the underlying surface, the output will be homotopy equivalent to the underlying surface.

One of the first works in the combinatorial surface model was the Dey and Schipper's implementation of Dehn's algorithm to test contractibility of a cycle [45], although edge weights are irrelevant in this result. Erickson and Har-Peled [58] prove that computing the shortest graph whose removal cuts a surface into a topological disk is NP-hard. Colin de Verdière and Lazarus consider the problem of finding the shortest *simple* loop [35] or cycle [34] within a given homotopy class in a combinatorial surface. A polynomial-time algorithm for the generalization of this problem to non-simple curves was recently obtained by Colin de Verdière and Erickson [33]. Erickson and Whittlesey [59] provide simple polynomial-time algorithms to compute the shortest homology basis and the shortest fundamental system of loops on a given surface.

Several authors have considered problem of computing a shortest noncon-

tractible or nonseparating cycle [58, 19, 15, 86], which we will discuss in Chapter 3. Algorithms to find shortest noncontractible or nonseparating cycles are used as subroutines in other algorithms on combinatorial surfaces [58, 33, 59] and in applications such as mesh simplification [124, 70], computing crossing numbers in graphs [81], low distortion probabilistic embeddings of graphs [79], and approximating TSP on high genus graphs [41]. Shortest noncontractible and nonseparating cycles are also examples of short tunnel and handle loops [43, 44], which are useful in topology repair, model editing, surface parameterization, and feature recognition.

The combinatorial surface model also has deep connections to results in graph theory. Kuratowski's theorem provides a characterization of planar graphs in terms of forbidden subdivisions [85]. Generalizations of this result show that graphs embedded on a surface of fixed genus have a finite forbidden minor characterization [103, 105]. In fact, for any fixed genus, we can test if a graph can be embedded on a surface of that genus in $O(n)$ time using this forbidden minor characterization [94]. In general, graphs embedded on surfaces of genus g require $\Omega(\sqrt{g})$ colors; however, if the smallest concontractible cycle is large, Thomassen showed that the graph can be colored with a constant number of colors [118]. Robertson and Seymour defined the face width, or *representivity*, of a graph embedded on a surface as the minimum number of times a noncontractible cycle intersects the graph [105]; in a combinatorial surface, this value is closely related to the length of the shortest noncontractible cycle. Large representivity in a graph leads to many other useful structural results; for example, large representivity means that a graph can be divided into small planar pieces [104].

The family of problems that we focus on in this thesis involve homotopy, which is a coarser form of classification that homeomorphism. Two maps $f : X \to Y$ and $g : X \to Y$ are *homotopic* if there is a continuous map $H : [0,1] \times X \to Y$ such that $H(0,x) = f(x)$ and $H(1,x) = g(x)$; we view H as a continuous deformation between f and g over the time interval $[0,1]$. In this thesis, we focus on homotopy of paths or cycles. For curves on surfaces, homotopy is equivalent to isotopy, or homotopy where every section $H(t,\cdot)$ is a homeomorphism; two simple curves are homotopic if and only if they are isotopic [5]. Homotopy is an equivalence relation on cycles based at a point; the fundamental group is the set of cycles based at a point under this equivalence relation where the group operation is concatenation, inverses are obtained by reversal, and the identity element is the class of contractible cycles. Deciding if a curve is contractible is thus equivalent to the word problem in a representation of the fundamental group. Conversely, for any finitely presented group G, one can construct a space whose fundamental group is G. Since the word problem for arbitrary groups is undecidable [98, 13], any algorithms to test contractibility or homotopy between curves must be in specific limited settings.

However, combinatorial surfaces are not an option when the underlying ob-

ject is not a manifold. Another type of topological space which we will discuss is a simplicial complex, or collection of simplices which are identified along common faces. Many algorithmic questions about manifolds are undecidable. Markov proved that determining if two 4-manifolds with a simplicial complex structure are homeomorphic is undecidable [90]; this work is based on the classical result that the word problem in groups is undecidable. On the other hand, Whittlesey showed that classifying finite 2-complexes up to homeomorphism is possible [121, 123, 122]; this was later shown to be equivalent to graph isomorphism, which is in NP but is not known to be NP-Complete or polynomially solvable [99]. The complexity of deciding if two 3-manifolds are homeomorphic is unknown; see [72] for a detailed discussion of results in this area. Markov's result [90] implies that even deciding if two paths are homotopic in a simplicial complex is undecidable.

For the sake of completeness, we briefly mention some results on the computation of homology groups, although the focus of this thesis is on homotopy. As previously mentioned, even simple homotopy questions are undecidable in general. As a result, many topological algorithms are based on *homology*, which provides a cruder classification of topological features than homotopy. For example, two cycles can be homologous on a surface but not homotopic; a noncontractible separating cycle is nullhomologous, even though it is not nullhomotopic.

Even in classical topology, determining homology groups is an algorithmic process. Tools such as exact sequences and Mayer-Viteoris sequences (see, e.g., [73]) are used to compute homology groups for many topological spaces and are very computational in nature. Homology groups are widely used in different application settings to gain information about the topological structure of an object [49, 22, 64, 50, 52, 63, 65, 125, 7, 8].

1.2 Our Contributions

In chapter 3, we describe an algorithm to compute the shortest noncontractible or nonseparating cycle in a combinatorial surface. The data structure which makes this algorithm possible computes the shortest path tree for every vertex along a single face of a combinatorial surface, generalizing a result of Philip Klein for planar graphs [82]. Our data structure maintains the shortest path tree kinetically [69] while the root of the tree moves along each edge of the face. Since shortest path trees are a commonly used algorithmic tool on graphs, potential applications for this results are numerous. This work is joint with Sergio Cabello, and a preliminary version appeared in SODA 2007 [16].

In chapter 4, we examine one generalization of planar separators [88] to graphs with higher genus. We show that computing the shortest *splitting* cycle, or noncontractible separating cycle, is NP-Hard, via a reduction from the Traveling Salesman Problem (TSP) in rectilinear graphs. We then give an algorithm to find the shortest splitting cycle in $g^{O(g)}n \log n$ time, where g is the genus of

graph and n is the size of the input. Our algorithm works by enumerating all possible crossing sequences of the shortest splitting cycle with a special family of shortest paths. This work is joint with Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey; it was first published at SoCG 2006 [26] and later appeared in CGTA [25].

In chapter 5, we look at a different measure of similarity between curves. Instead of considering the homotopy type of a cycle, as in [33], we wish to consider two curves which are known to be homotopic and find the "smallest" homotopy between them, which we call the *homotopic Fréchet distance*. When the curves are on surfaces with positive curvature, such as is the case with many combinatorial surfaces, algorithms to compute this value seem to be quite difficult. We examine this problem for the Euclidean plane minus obstacles, giving a polynomial-time algorithm, and characterize the type of relative homotopy classes which are possible in surfaces with non-positive curvature. Our result is closely related to other algorithms which look for shortest paths in specified homotopy classes of the plane minus obstacles [55, 9, 76] and finding shortest paths on surfaces of positive curvature such as convex polyhedra [93]. This work is joint with Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite, and a preliminary version appeared in SoCG 2008 [24].

In chapter 6, we look at homotopy questions in a type of simplicial complex called the *Viteoris-Rips complex*, or Rips complex. We characterize the fundamental group of such a complex when the point set is a subset of the plane, and briefly describe related results on algorithms to test contractibility of cycles and compute the shortest noncontractible cycle in such complexes [27]. We also examine the fundamental group of *quasi-Rips complexes*, a generalization of standard Rips complexes that model noisy data or uncertainty. This work is joint with Jeff Erickson, Rob Grist, and Vin de Silva.

Chapter 2

Definitions and Background

2.1 Topological Definitions

A *surface* (or 2-manifold with boundary) $\mathcal{M}$ is a topological Hausdorff space where each point has a neighborhood homeomorphic either to the plane or to the closed half-plane. The points with neighborhood homeomorphic to the closed half-plane comprise the *boundary* of $\mathcal{M}$. All the surfaces considered here are *compact* and *connected*. A surface is nonorientable if it contains a topological Möbius-strip, and otherwise it is orientable. Any orientable surface is homeomorphic to a sphere with g handles attached and b open disks removed, while any nonorientable surface is homeomorphic to the connected sum of g projective planes with b open disks removed. In both cases, we refer to g as the *genus* of the surface, and to b as the number of *boundaries*.

Let Σ be a surface. A *path* on Σ is a continuous map $p : [0,1] \to \Sigma$. A *cycle* is a continuous map $\gamma : S^1 \to \Sigma$, where S^1 denotes the unit circle. A *loop* with *basepoint* x is a path p with $x = p(0) = p(1)$. An *arc* α is a path whose endpoints are in the boundary. (We will use *curve* as a generic term for paths, loops, cycles, and arcs.) Any curve is *simple* if it is one-to-one, except at the basepoint in the case of loops. Two curves are *disjoint* if they do not intersect. The notation $\alpha + \beta$ is used for the concatenation of curves α and β, assuming that endpoints or basepoints match accordingly.

If S is a set of pairwise disjoint simple curves, $\mathcal{M} \setminus S$ denotes the surface with boundary obtained by *cutting* $\mathcal{M}$ along the loops or arcs in S. A simple cycle γ is *separating* if $\mathcal{M} \setminus \gamma$ has two components.

A *homotopy* between two paths p and q is a continuous map $H : [0,1] \times [0,1] \to \Sigma$ such that $H(0,\cdot) = p(\cdot)$ and $H(1,\cdot) = q(\cdot)$. Note that endpoints remain fixed during a homotopy. A (free) *homotopy* between two cycles γ and δ is a continuous map $H : [0,1] \times S^1 \to \Sigma$ such that $H(0,\cdot) = \gamma$ and $H(1,\cdot) = \delta$. Two curves β and γ are *homotopic* when there is some homotopy between them, and we denote it by $\beta \sim \gamma$.

Up to homotopy, each boundary component of Σ admits two parameterizations as simple cycles, one being the reverse of the other. We say that a cycle is homotopic to a boundary component of Σ when it is homotopic to either of these two parameterizations. If γ is a simple cycle homotopic to a boundary

δ of Σ, then $\Sigma \setminus \gamma$ has two connected components, one of them a topological annulus with γ and δ as boundaries.

A cycle is *contractible* if it is homotopic to a constant cycle, or point. An arc whose endpoints are in the same boundary component δ is noncontractible if it is not homotopic to a subpath of δ. Every contractible simple cycle is separating, since it bounds a disk, but not all simple separating cycles are contractible. We say a cycle is *splitting* if it is simple, noncontractible, and separating.

It is not hard to verify that homotopy is an equivalence relation on paths and loops. The *fundamental group* $\pi_1(X, x_0)$ for a topological space X and point $x_0 \in X$ is the group consisting of homotopy classes of loops based at x_0 with concatenation of loops as its group operation. Any set of $2g$ loops which generate $\pi_1(M, m)$ are called a *homotopy basis*; we frequently refer to a loop in such a basis as a *generator*.

Let $\mathcal{M}$ be a surface with genus g and with b boundary components. If $b = 0$, a *system of loops* on $\mathcal{M}$ is a set of pairwise disjoint simple loops L with a common basepoint such that $\mathcal{M} \setminus L$ is a topological disk. Any system of loops contains exactly $2g$ loops. $\mathcal{M} \setminus L$ is a $4g$-gon where each loop appears as two boundary edges; this $4g$-gon is called the *polygonal schema* associated with L.

If $b \geq 1$, a *system of arcs* on $\mathcal{M}$ is a set of pairwise disjoint simple arcs A such that $\mathcal{M} \setminus A$ is a topological disk. Any system of arcs contains exactly $2g + b - 1$ arcs, by Euler's formula and standard double-counting argument. $\mathcal{M} \setminus A$ is a $(8g + 4b - 4)$-gon where each arc appears as two boundary edges, the remaining $4g + 2b - 2$ edges of the $(8g + 4b - 4)$-gon corresponding to pieces of the boundaries of $\mathcal{M}$; this $(8g + 4b - 4)$-gon is called the *polygonal schema* associated with A.

For any topological space X, a *covering space* is a topological space $\hat{X}$ together with a surjective map $p : \hat{X} \to X$, such that for every point $x \in X$ there is an open neighborhood $x \in U \subseteq X$ such that $p^{-1}(U)$ is a disjoint union of sets in $\hat{X}$ which are homeomorphic to U under p. The *universal cover* is the unique covering space that is simply connected, meaning the fundamental group is trivial.

Two cycles are *homologous* (with $\mathbb{Z}_2$ coefficients) if one can be continuously deformed into the other via a deformation that may include splitting cycles at self-intersection points, merging intersecting pairs of cycles, or adding or deleting contractible cycles. Thus, if two cycles are homotopic, they are also homologous, but the converse is not necessarily true. A cycle or loop is *null-homologous* if it is homologous to a constant loop. A simple cycle γ is null-homologous if and only if it is *separating*, that is, if $\mathcal{M} \setminus \gamma$ has two components. Every contractible simple cycle is separating (in fact, it bounds a disk), but not all simple cycles are contractible.

Let M be a simplicial complex. Given an artibrary ring R, a *k-chain* is a linear combination of oriented k-simplices with coefficients from R. The *k^{th} chain group* C_k is the free abelian group of k-chains. The *boundary operator*

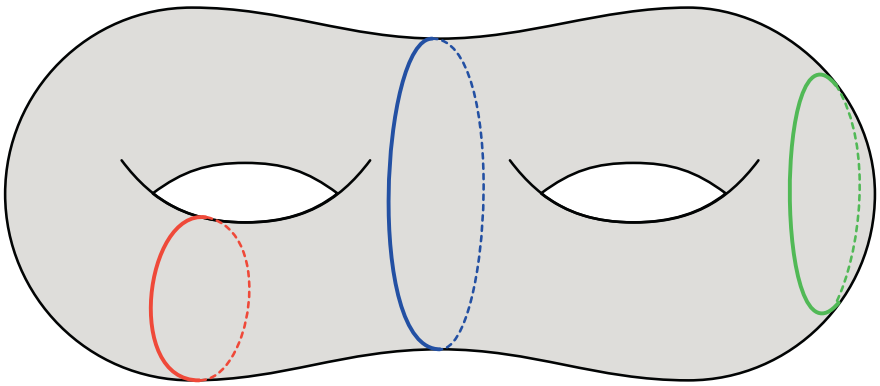

Figure 2.1. The leftmost cycle is nonseparating and noncontractible, the center cycle is separating and noncontractible, and the rightmost cycle is separating contractible.

$\partial_k : C_k \to C_{k-1}$ is a linear map which takes each (oriented) k-simplex to the sum of its (oriented) $(k-1)$-facets. A *k-cycle* is a k-chain whose boundary is empty; the set of k-cycles is the kernel of ∂_k, and so forms a subgroup of C_k. A *k-boundary* is a k-chain which is the boundary of some $(k+1)$-cycle; the set of k-boundaries is the image of ∂_{k+1} and so also forms a subgroup of C_k. The k^{th} *homology group* $H_k(M, R)$ is the abelian group consisting of k-cycles quotiented out by the k-boundaries. The k^{th} *Betti number* is defined to be the rank of H_k.

2.2 Combinatorial Surfaces

A combinatorial surface is a undirected, weighted graph $G(\mathcal{M})$ embedded on a surface of genus g so that each face of the graph is a topological disk on $\mathcal{M}$. Curves on this surface are required to be walks on $G(\mathcal{M})$, and edges of $G(\mathcal{M})$ have positive weights, allowing one to measure the length of a curve. Note that we allow edges to be used multiple times on a simple path or cycle as long as the path or cycle does not actually cross itself; in other words, a curve is *simple* if it can be infinitesimally perturbed to a simple curve in $\mathcal{M}$. This is a notable difference from the standard graph theoretical definition of a simple path, where no edge can appear more than once. We use $|\alpha|$ to denote the length of a curve α. The multiplicity of a path is the maximum number of times that an edge appears in it.

We use n for the number of vertices plus the number of edges in the graph, and the weight of an edge uv is denoted $w(uv)$. We use $V = V(G)$, $E = E(G)$, and $F = F(G)$ to denote the set of vertices, edges, and faces of G, respectively. The *dual (multi)graph* of G, written G^*, is a graph formed by making every face of G a vertex and adding edges between adjacent faces. G^* has a natural embedding in Σ: for $e \in E(G)$, we use e^* to denote the edge in the dual graph which goes between the two faces that e borders. For a set of edges $D \subseteq E$ we use $D^* = \{e^* \mid e \in D\}$.

Euler's formula asserts that

$$|V| - |E| + |F| = \chi(\Sigma)$$

where $\chi(\Sigma)$ is the Euler characteristic of Σ, which is $2-2g$ for orientable surfaces and $2-g$ for nonorientable. Since we assume a simple graph, each face has at least 3 edges, and therefore $2|E| \leq 3|F|$. It follows then from Euler's formula that G has $O(|V| + g)$ edges.

It is often more convenient to work in an equivalent dual formulation of this model introduced by Colin de Verdière and Erickson [33]. A *cross-metric surface* is also an abstract surface $\mathcal{M}$ together with an undirected weighted graph $G^* = G^*(\mathcal{M})$, embedded so that every open face is a disk. However, now we consider only *regular* paths and cycles on $\mathcal{M}$, which intersect the edges of G^* only transversely and away from the vertices. The *length* of a regular curve p is defined to be the sum of the weights of the dual edges that p *crosses*, counted with multiplicity. See [33] for further discussion of these two models.

Many types of cycles can be computing using Thomassen's *3-path condition* [95]. A set of cycles C satisfies the 3-path condition if for any pair of vertices u and v and disjoint paths p_1, p_2, and p_3 with endpoints u and v, if two of the three cycles formed by concatenating the paths are not in C, then the third cycle cannot be in C. This condition leads to an polynomial time algorithm to find the smallest cycle of the set C, assuming that we can test membership in C in polynomial time.

To simplify many of our proofs and algorithms, we will assume that the lengths of shortest paths are unique. To enforce this, we add random infinitesimal weights to each edge; the Isolation Lemma [97] then implies that the lengths of all shortest paths are unique with high probability.

Chapter 3

Shortest Noncontractible or Nonseparating Cycles

In this chapter, we develop an algorithm to find a shortest noncontractible cycle in a combinatorial surface in $O((b + g^3)n \log n)$ time, and a shortest nonseparating cycle in $O(g^3 n \log n)$ time. Several authors have previously considered problem of computing a shortest noncontractible or nonseparating cycle. The first algorithm to compute these cycles relied upon the so-called *3-path condition* [95]; this algorithm finds a shortest noncontractible or nonseparating cycle in $O(n^3)$ time [95, Sect. 4.3], regardless of the genus of the graph. Erickson and Har-Peled describe a faster algorithm to compute nonseparating and noncontractible cycles time in $O(n^2 \log n)$ time, again regardless of the genus of the graph [58]. Cabello and Mohar [19] gave an algorithm which runs in time $g^{O(g)} n^{3/2} \log n$, the first algorithm for this problem to have a running time which was bounded by a function of the genus. Cabello [15] later improved the running time using separators to $g^{O(g)} n^{4/3}$. Kutz [86] developed an algorithm with a running time of $g^{O(g)} n \log n$ which computed a finite portion of the universal cover and then did standard shortest path computations in the resulting planar graph.

The main tool we use for finding shortest noncontractible or nonseparating cycles is a data structure that quickly computes shortest paths from all vertices on a single face of an embedded graph. A *shortest path tree* in a graph is a tree containing shortest paths from a specified vertex to all other vertices. Shortest path trees are a fundamental tool on graphs, and have applications for flows, distance queries, connectivity, and many other problems.

In the planar graph setting, there are many results for multiple source shortest paths. The result of primary interest is by Klein [82], who addressed the problem of maintaining the shortest path tree in a planar graph as the source of the tree moves along some face in the graph. He gave an algorithm that computed an implicit representation of the shortest path tree for all vertices on a common face in the graph in $O(n \log n)$ time; shortest path queries to any vertex on the face then take $O(\log n)$ time. Other noteworthy results on multiple source shortest paths for planar graphs include Frederickson's all-pairs shortest paths representation [61], Lipton and Tarjan's planar separator theorem [88], and Schmidt's $O(n \log n)$ algorithm that supports distance queries for specific subsets of vertices on a grid [106].

In this chapter, we develop an algorithm to maintain the shortest path tree as the source of the tree moves around a face of a graph embedded on a surface of genus $g \geq 1$. The running time to compute the shortest path tree for every vertex on a given face is $O(g^2 n \log n)$. If we make the underlying kinetic data structure persistent [47], shortest path queries take $O(\log n)$ time. To contrast this with previously known results, note that a graph with n vertices embedded in a surface of genus g has $O(g+n)$ edges by Euler's formula, and therefore we can solve the all-pairs shortest path problem in $O(n(n \log n + g + n)) = O(gn + n^2 \log n)$ time using Dijkstra's algorithm with Fibonacci heaps [62]. Our result improves this running time when $g = o(\sqrt{n})$. We can thus restrict our attention to the case $g \leq n$, in which case G has $O(n)$ edges.

Klein's algorithm begins with a shortest path tree rooted at a vertex on the specified face. It declares the root of the tree to be at a neighbor of the original root. The tree is no longer necessarily a shortest path tree, so edges are iteratively added and removed from the tree from on a set of candidate edges which could belong to the shortest path tree. The next edge chosen is what Klein calls the leafmost unrelaxed edge; this characterization is based on the fact that the set of edges not present in the tree form a tree in the dual graph, called a *co-tree*. Klein shows how to find these edges quickly and proves that no edge will be added or removed from a shortest path tree more than a constant number of times.

The main obstacle to extend Klein's algorithm to genus g graphs is that the complement of a tree is no longer a co-tree, so there is no "leafmost unrelaxed edge". After moving the root to a neighboring vertex, it is difficult to quickly determine which edges to add and remove from the original shortest path tree and in what order they should be added.

Our approach instead maintains a *kinetic* shortest path tree. For more background on kinetic data structures in general, see [69]. Conceptually, we move the root of the tree *continuously* along the edges of the specified face. As the root moves, edges will appear and disappear from the shortest path tree at discrete events; our algorithm efficiently computes these events and updates the tree. The key fact for our analysis is that the time used to move the source along an edge depends on the symmetric difference of the starting and final shortest path trees. We show that the total number of times an edge can enter or leave the shortest path tree as the root moves around a face is a function of the genus of the underlying surface.

3.1 Definitions and Tools

3.1.1 Shortest Path Trees

Consider a rooted tree T spanning a graph G with weighted edges, and let $s$denote its root (or source). We define a function $d_T : V(T) \to \mathbb{R}$ as the

distance from $s$to a given vertex. We omit the subscript T when it is clear from context. Consider an edge uv from G. The *tension* of the directed edge $\overrightarrow{uv}$ is defined as $t(\overrightarrow{uv}) = d(v) - w(\overrightarrow{uv}) - d(u)$. We say that the edge $\overrightarrow{uv} \in A(\overrightarrow{G})$ is *tense* if $t(\overrightarrow{uv}) > 0$. In other words, if $\overrightarrow{uv}$ is tense, there is a shorter path from $s$to v which uses the path in T to u plus the edge uv, rather than the (unique) path from $s$to v contained in T. We say that an (undirected) edge $uv \in E(G)$ is *tense* whenever either $\overrightarrow{uv}$ or $\overrightarrow{uv}$ is tense. It is a simple exercise to verify that if T leaves no tense edges in G, then T is a shortest path tree. For more background on this topic, see [114].

Although the graph G is undirected, we actually maintain a shortest path tree in the directed graph $\overleftrightarrow{G}$, where each edge of G appears twice. Shortest path tree edges are directed towards from the root s. The running time of some results will be expressed as a function of $|T \setminus T'|$ for shortest path trees T and T'. Assuming the roots of T and T' are distinct, this quantity is at least 1, since $T \setminus T'$ must contain a directed edge connecting the roots.

3.1.2 Dynamic Forests

Our algorithms require dynamic forest data structures that implicitly maintain edge or vertex values under edge insertions, edge deletions, and updates to the values in certain subtrees. We use two known data structures, whose interface we describe next.

Vertex Values

The first data structure maintains vertex-values in a dynamic rooted directed forest under the following operations:

- CREATE(val) adds a tree with a single vertex of value val to the forest.
- CUT(e) removes the edge e from the forest. The subtree that contains the original root $s$keeps $s$as it root, while the other subtree takes as root the tail of e.
- JOIN(u, v) adds the edge uv to the forest and sets the root of the subtree containing u as root of the tree containing the new edge. Note that JOIN(u, v) and JOIN(v, u) give rise to the same tree but with different roots. This operation assumes that u and v are initially in different trees.
- GETVALUE(v) returns the value stored at the vertex v.
- ADDSUBTREE(Δ, v) adds the value Δ to every vertex in the subtree rooted at v.
- SAMESUBTREE(u, v) returns true if u is in the subtree rooted at v.

Using Euler-Tour trees [75], each operation takes $O(\log n)$ amortized time. See Tarjan [115] for a precise discussion.

Edge Values

The second data structure maintains $\textsc{Val}(e)$ for each edge e in a dynamic forest. Each value is an ordered pair (t_1, t_2). The following operations will be necessary in our algorithm.

- $\textsc{Create}()$ adds a tree with with a single vertex to the forest.
- $\textsc{Cut}(e)$ removes the edge e from the tree T that contains it. The subtree that contains the original root $s$of T keeps $s$as it root, while the other subtree takes as root the endpoint of e that it contains.
- $\textsc{Join}(u, v, \textsc{Val}_1, \textsc{Val}_2)$ adds the edge uv to the forest, sets the root of the old tree containing u as the root of the tree containing uv, and sets $t_1(uv) = \textsc{Val}_1$ and $t_2(uv) = \textsc{Val}_2$. Note that $\textsc{Join}(u, v, val_1, val_2)$ and $\textsc{Join}(v, u, val_1, val_2)$ give rise to the same tree but with different roots. (This operation assumes that u and v are initially in different trees.)
- $\textsc{Lca}(u, v)$ finds the lowest common ancestor of vertices u and v. (This operation assumes that u and v are in the same tree.)
- $\textsc{GetValue}(e, i)$ returns the value $t_i(e)$.
- $\textsc{AddPath}(\Delta, u, v, i)$ adds Δ to $t_i(e)$ for each edge e in the (unique) path from vertex u to vertex v. (This operation assumes that u and v are in the same tree.)
- $\textsc{MaxPath}(u, v, i)$ finds the edge e with largest value $\textsc{Val}_i(e)$ in the (unique) path from vertex u to vertex v. (This operation assumes that u and v are in the same tree.)
- $\textsc{Swap}(u, v)$ swaps the values $\textsc{Val}_1(e)$ and $\textsc{Val}_2(e)$ for each edge e in the (unique) path from vertex u to vertex v. (This operation assumes that u and v are in the same tree.)

To perform these operations, we use self-adjusting top trees [116], which can perform each of the operations in $O(\log n)$ amortized time. Each operation is a standard operation for self adjusting top trees. We briefly describe self adjusting top trees below, and then describe how the operations above are implemented.

Self-adjusting top trees [116] maintain an n-vertex forest that can add or remove edges between the vertices and stores data associated with each edge or vertex. Updates to the stored data can happen to entire paths or subtrees. The ability to update paths (as well as subtrees) quickly is key to our algorithm. The amortized running time for any of these operations - `join`, `cut`, or `update` - is $O(\log n)$.

Self-adjusting top trees decompose any tree on a graph with two operations: rake and compress. These operations were originally proposed by Miller and Reif [92]. A *rake* takes a degree one vertex and removes it, essentially collapsing

that edge onto a neighboring edge. A *compress* takes a degree two vertex and removes it, replacing it with a single edge. See Figure 3.1.

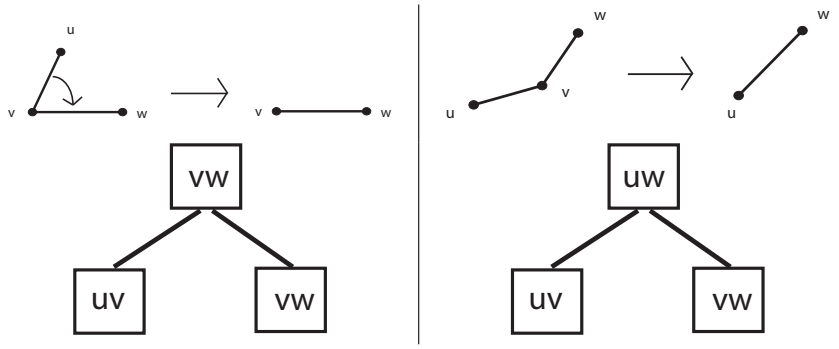

Figure 3.1. Left: A rake operation and the resulting tree created. Right: A compress operation and the resulting tree.

In top trees [1], a series of rake and compress operations are used to reduce the input forest to a single edge. To form a tree based on this decomposition, make each edge of the tree a leaf in the data structure, and create a parent node above any two edges in a rake or compress operation. For each rake operation taking an edge uv and raking it onto vw, promote vw to be the name of the parent node. For each compress operation with the edges uv and vw, label the parent node uw to represent the new edge created. See Figure 3.1. This gives an $O(n)$ size tree data structure which records the decomposition of the input tree. Insertions, deletions, and updates to subtrees can be performed in $O(\log n)$ time.

In order to allow path updates, self-adjusting top trees [116] decompose the tree into paths before doing rake or compress operations. All edges are oriented towards a root vertex. Then a path from some leaf to the root is chosen as the top level path; this path is processed using compress operations. All other components of the tree, which are subtrees rooted at vertices along the top level path, are recursively processed using rake and compress operations. Then these single edges can be raked onto edges in the top level path.

In self-adjusting top trees, the key new operation is EXPOSE. The EXPOSE operation changes this decomposition so that the specified vertices are the endpoints of the top level path in the tree. This operation is executed by alternating splay and splice operations, which run in amortized $O(\log n)$ time. As a result, EXPOSE and updates or queries to both subtrees and paths in the forest take amortized $O(\log n)$ time.

For our algorithm, the operations CREATE, CUT, JOIN, and GETVALUE are identical to those operations in the self-adjusting top tree. ADDPATH and MAXPATH can be implemented using EXPOSE along with the standard data manipulation operations. LCA can be accomplished by calling EXPOSE on the two endspoints and then finding their least common ancestor on that path. SWAP can be implemented by calling GETVALUE for each of the two values,

cutting the edge, and then calling join with the two values swapped.

3.2 The Data Structure

3.2.1 Preliminaries

Suppose that we are given the shortest path tree T rooted at some vertex u. We wish to modify the tree so that it becomes the shortest path rooted at v, a neighbor of u. View the shortest path tree as a kinetic data structure, in which the root $s$slides continuously from u to v along the edge $e = uv$. Any vertex in the subtree of T rooted at u is called a *blue* vertex. All other vertices are colored *red*. Note that we are labeling each vertex red or blue depending on whether its distance to $s$is increasing (red) or decreasing (blue). The vertices of each color form a subtree of T, since any vertex's shortest path to $s$must go through either u or v. The blue subtree of T is a subtree of the shortest path tree rooted at v, because blue shortest paths do not change as $s$moves closer to v. Eventually, when $s$arrives at v, the shortest path tree rooted at v is obtained.

Suppose $s$slides a distance of Δ across the edge e. The distance from $s$to every red vertex increases by Δ, and the distance from $s$to every blue vertex decreases by Δ. As long as no edge in $G \setminus T$ is tense, T is still the shortest path tree. Consider the tension of any edge. Any edge with monochromatic endpoints (including edges in T) has constant tension, since the distance from its endpoints to the root are changing at the same rate. Therefore, any edge whose tension is changing must have one blue endpoint and one red endpoint. Call this set of edges *green*.

When a directed edge $\overrightarrow{xy}$ is about to become tense (i.e. when $t(\overrightarrow{xy})$ is increasing and $t(\overrightarrow{xy}) = 0$), then $d(y) = d(x) + w(\overrightarrow{xy})$ for some red vertex y and blue vertex x. This means that as $s$continues to move along e, the path from $s$to $s$that goes through x is about to become shorter than the current path in the shortest path tree. When an edge $e = xy$ is about to become tense, we have an *event* in the shortest path tree. At each such event, the edge directed out from y to $s$is deleted, and the edge $\overrightarrow{xy}$ is added to the shortest path tree instead. Additionally, y and all the vertices in its subtree are recolored to blue, since the shortest path to $s$now passes through v last. The set of green edges also changes and needs to be updated.

In our data structure, the search for events reduces to finding the (green) edge which will become tense first. Each time such an event occurs, we perform one Cut and one Join in the shortest path tree in $O(\log n)$ amortized time. The main remaining issue is detecting which green edges become tense.

Our algorithm uses two data structures, one for the shortest path tree T and the other for the complementary dual graph $(G \setminus T)^*$. The main difference between the algorithms for the planar case and the higher genus case is the dual structure. In a planar graph, $(G \setminus T)^*$ is a spanning tree of the dual graph, called

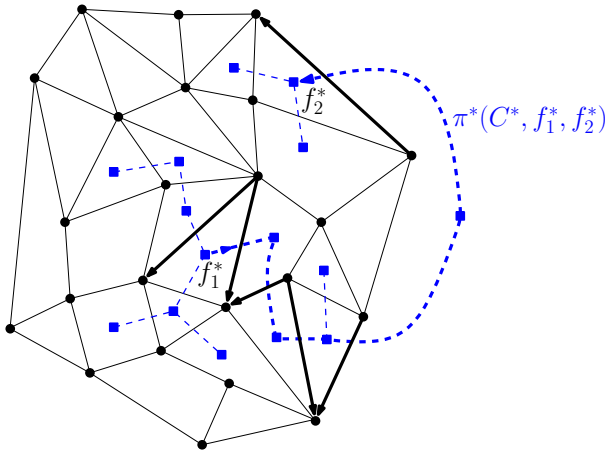

Figure 3.2. A planar graph drawn with solid (black) segments. A tree in the dual C^* is with dashed (blue) segments. The path $\pi(C, f^*, g^*)$, whose orientation is indicated with two arrows, is depicted with thick dashed (blue) segments, while the black thick arcs correspond to the arcs $\overrightarrow{LR}(C, f^*, g^*)$.

a *cotree*. However, in a genus g graph, $(G \setminus T)^*$ is a cotree plus $2g$ additional edges.

We maintain the shortest path tree T is stored using a vertex-valued dynamic forest structure as described in 3.1.2; the value at each node is its distance in T from the root. We maintain $(G \setminus T)^*$ using the edge-labeled dynamic forest data structure described in 3.1.2; the value stored for each edge is the tensions of $\overrightarrow{xy}$ and $\overrightarrow{yx}$. In the next section, we describe the dual data structure in more detail.

3.2.2 Tensions of Dual Trees in Orientable Surfaces

We restrict our attention to orientable surfaces. Let C be a subtree of G^*. For simplicity, we assume C is undirected. For any two nodes f^* and g^* of C (dual to faces f and g in G), let $\pi(C, f^*, g^*)$ be the unique directed path in C from f^* to g^*. Let $\overrightarrow{LR}(C, f^*, g^*)$ be the (directed) edges of $\overleftrightarrow{G}$ that cross $\pi(C, f^*, g^*)$ from left to right, that is,

$$\overrightarrow{LR}(C, f^*, g^*) = (\pi(C, f^*, g^*))^*$$

and similarly define

$$\overleftarrow{RL}(C, f^*, g^*) = (\pi(C, f^*, g^*))^*$$

See Figure 3.2. Note that $\overrightarrow{LR}(C, f^*, g^*) = \overleftarrow{RL}(C, g^*, f^*)$.

We dynamically maintain a collection of dual trees $C_1, C_2, \ldots$ whose union is $(G \setminus T)^*$. Each tree C_i stores the tensions $t(\overrightarrow{xy})$ and $t(\overrightarrow{yx})$ for every dual edge $(xy)^*$ in C_i. The structure supports the following operations:

- CUT(e) removes the edge e from the tree that contains it. (This operation

assumes that e is an existing edge in some tree.)

- JOIN(u, v, t_1, t_2) takes two primal vertices u and v and adds the edge uv^*, and then sets tensions $t(\overrightarrow{uv}) = t_1$ and $t(\overrightarrow{vu}) = t_2$ (This operation assumes that uv^* connects two distinct dual trees.)

- MAXTENSION$\overrightarrow{LR}(f^*, g^*)$ returns an edge $xy \in E(G)$ maximizing the value $t(\overrightarrow{xy})$ among the arcs $\overrightarrow{xy} \in \overrightarrow{LR}(C, f^*, g^*)$, where C is the tree containing nodes f^* and g^*. (This operation assumes that f^*, g^* are nodes of a single tree.)

- MAXTENSION$\overleftarrow{RL}(f^*, g^*)$ returns an edge $xy \in E(G)$ maximizing the value $t(\overrightarrow{xy})$ among the arcs $\overrightarrow{xy} \in \overleftarrow{RL}(C, f^*, g^*)$, where C is the tree containing nodes f^* and g^*. (This operation assumes that f^*, g^* are nodes of a single tree.)

- ADDTENSION$\overrightarrow{LR}(f^*, g^*, \Delta)$ adds the value $+\Delta$ to the tension $t(\overrightarrow{xy})$ for all $\overrightarrow{xy} \in \overrightarrow{LR}(C, f^*, g^*)$, where C is the tree containing nodes f^* and g^*. (This operation assumes that f^* and g^* are nodes of a single tree.)

- ADDTENSION$\overleftarrow{RL}(f^*, g^*, \Delta)$ adds the value $+\Delta$ to the tension $t(\overrightarrow{xy})$ for all $\overrightarrow{xy} \in \overleftarrow{RL}(C, f^*, g^*)$, where C is the tree containing nodes f^* and g^*. (This operation assumes that f^* and g^* are nodes of a single tree.)

- JUNCTION(f^*, g^*, h^*) returns $\pi(f^*, g*) \cap \pi(g^*, h^*) \cap \pi(f^*, h^*)$, the unique node common to the 3 paths connecting any pair from $\{f^*, g^*, h^*\}$.

Lemma 3.2.1. *There is a data structure to store a dynamic collection of dual trees where the operations* CUT, JOIN, MAXTENSION$\overrightarrow{LR}$, MAXTENSION$\overleftarrow{RL}$, ADDTENSION$\overrightarrow{LR}$, ADDTENSION$\overrightarrow{LR}$, *and* JUNCTION *each take* $O(\log n)$ *time.*

Proof: For each dual tree C_i, choose one of its nodes and set it as the root. Each rooted tree C_i is then stored in an edge valued dynamic forest structure, as described in Section 3.1.2. For convenience, we direct each tree towards its root, although the underlying tree is still undirected. The two values of a dual edge are the two associated tensions $t(\overrightarrow{xy})$ and $t(\overrightarrow{yx})$, according to the following correspondence:

(1) if xy^* points towards the root of its tree and $\overrightarrow{xy}$ crosses xy^* from left to right, then $\text{VAL}_1(xy^*) = t(\overrightarrow{xy})$ and $\text{VAL}_2(xy^*) = t(\overrightarrow{yx})$. See Figure 3.3.

We will be careful to keep this correspondence through the operations of the data structure.

The operation CUT(e^*) is implemented by calling CUT(e^*) in the underlying self adjusting top tree. Correspondence (1) is maintained in the two new trees, because the new tree's root is the remaining endpoint of e^*.

Now consider JOIN(u, v, t_1, t_2). Let f_ℓ and f_r be the faces to the left and the right of $\overrightarrow{uv}$, respectively, and let root(f_ℓ^*) and root(f_r^*) be the roots of

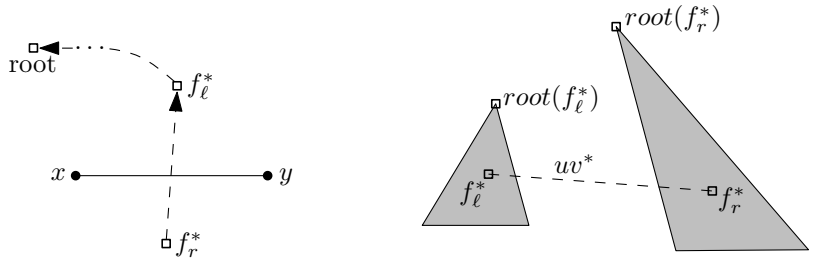

Figure 3.3. Figures for Lemma 3.2.1. Left: Notation for correspondence (1). Right: Schematic description of the operation JOIN.

the dual trees containing f_ℓ^* and f_r^*, respectively. See Figure 3.3, right. We call $\textsc{Join}(f_\ell^* f_r^*, t_1, t_2)$ in the self-adjusting top tree. The resulting cotree has $\text{root}(f_r^*)$ as its root. Correspondence (1) holds for all edges except those in the path from $\text{root}(f_r^*)$ to f_ℓ^*. We fix this inconsistency by calling $\textsc{Swap}(\text{root}(f_r^*), f_\ell^*)$.

We now consider $\textsc{MaxTension}\overrightarrow{LR}(f_1^*, f_2^*)$. Let C be the tree containing the nodes f^* and g^*, and let a^* be the lowest common ancestor in C of f^* and g^*, found by calling $\textsc{Lca}(f^*, g^*)$. The path $\pi(C, f^*, g^*)$ consists of an "ascending" path π_{up} from f^* to a^*, followed by a "descending" path π_{down} from a^* to g^*. Because of correspondence (1), an edge along the path π_{up} store the relevant tension as $\textsc{Val}_1(\cdot)$, while along π_{down} this tension is stored as $\textsc{Val}_2(\cdot)$. To find the edge with maximum tension, we call $\textsc{Lca}(f^*, g^*)$ the self adjusting top tree to find a^* and then call $\textsc{MaxPath}(f^*, a^*, 1)$ and $\textsc{MaxPath}(a^*, g^*, 2)$ and return the maximum of the two values returned.. Note that $\textsc{MaxTension}\overleftarrow{RL}(f^*, g^*)$ is an equivalent operation.

For $\textsc{AddTension}\overrightarrow{LR}(f^*, g^*, \Delta)$, we again first find $a^* = \textsc{Lca}(f^*, g^*)$ the self adjusting top tree containing f^* and g^*. We then update the tensions in each monotone subpath separately by performing $\textsc{AddPath}(\Delta, f^*, a^*, 1)$ and $\textsc{AddPath}(\Delta, a^*, g^*, 2)$. Note that $\textsc{AddTension}\overleftarrow{RL}(f^*, g^*, \Delta)$ is an equivalent operation.

Finally, the operation $\textsc{Junction}(f^*, g^*, h^*)$ can be performed with a constant number of LCA queries, since the junction must be the lowest common ancestor of two of the three input nodes.

Since each operation requires a constant number of calls to the self adjusting top tree, each operation takes $O(\log n)$ amortized time.

□

3.3 Planar Graphs

3.3.1 Algorithm for Planar Graphs

Our algorithm for planar graphsis similar to Klein's algorithm [82] in that we relax edges and use a tree/cotree decomposition. However, we relax edges in a difference order than Klein's algorithm; our kinetic data structure relaxes edges one by one as the source moves, instead of changing the root and then relaxing

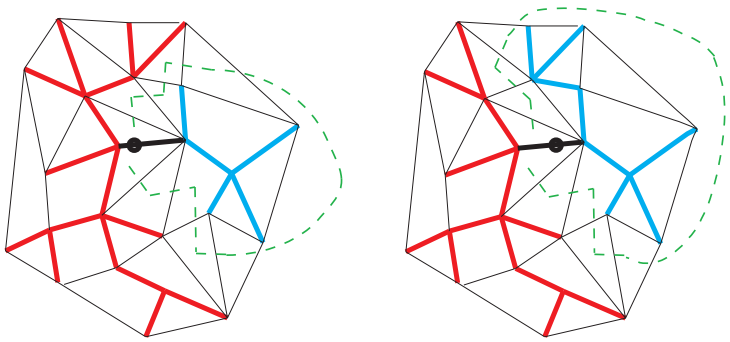

Figure 3.4. The thick lines in the graph are the shortest path tree rooted at s, shown moving along the edge e. As $s$moves closer to the target, an edge along the green path in the dual (shown in dotted lines) becomes tense and is inserted.

edges iteratively from a set of tense edges as Klein does.

Let T be a shortest path tree rooted at a point $s$along the edge $e = uv$, and let f_r, f_ℓ denote the faces to the left and to the right of $\overrightarrow{uv}$, respectively. Recall that vertices in the subtree rooted at u are red, and vertices in the subtree rooted at v are blue. Any edge between red and blue vertices is called green; this is the set of edges whose tension is changing.

Initially, if the edge e is not in the shortest path tree, all vertices are red. At some stage e becomes tense, since the distance from $s$to v along e is going to zero as $s$slides along e. At this stage, the edge e enters the shortest path tree, and the subtree rooted at v is immediately colored blue.

Once e is in T, we are sliding the root $s$of the shortest path tree T across uv; we must maintain a shortest path tree as the root moves. The dual edges $(G \setminus T)^*$ form a tree C of the dual graph, and the green edges are those dual to the edges in $\pi(C, f_r^*, f_\ell^*)$. The blue vertices lie in the region to the left of the cycle $\pi(C, f_r^*, f_\ell^*)$ concatenated with uv^*, while the red vertices are to the right. See Figure 3.4. Therefore, the directed edges $\overrightarrow{LR}(C, f_r^*, f_\ell^*)$ are oriented from blue to red vertices, and their tensions are increasing as $s$slides across e. Symmetrically, the directed edges $\overleftarrow{RL}(C, f_r^*, f_\ell^*)$ have decreasing tensions. Any directed edge not in $\overrightarrow{LR}(C, f_r^*, f_\ell^*)$ or $\overleftarrow{RL}(C, f_r^*, f_\ell^*)$ has constant tension since both endpoints have the same color.

Our algorithm begins with a shortest path tree T rooted at a vertex $r = u$. The dual edges $(G \setminus T)^*$ form a spanning tree T^* of the dual graph, which we store in an Euler tour tree as described in Section 3.2.2.

Once $e = uv$ is in the shortest path tree, the algorithm iterates over the following steps until $s$reaches v or e leaves the shortest path tree (at which point every vertex is blue). See Figure 3.4. We call MaxTension$\overrightarrow{LR}(f_r^*, f_\ell^*)$ to find the edge $\overrightarrow{xy}$ with maximum tension. Since $t(\overrightarrow{xy})$ is increasing, this will be the first edge to become tense. Let $\Delta = t(\overrightarrow{xy})/2$ be the amount the root needs to move for $\overrightarrow{xy}$ to become tense.

Next, move the root sa distance Δ along uv, update distances in the primal tree, and update tensions in the dual tree. In the primal tree, AddSubtree(u, Δ)

adds Δ to every distance for vertices in the red subtree, simulating the root sliding along the edge e by a distance of Δ. Similarly, AddSubtree$(v, -\Delta)$ updates the distance from $s$to the blue vertices. In the dual structure, we update the tensions of the green edges by calling AddTension$\overrightarrow{LR}(2\Delta, f_r^*, f_\ell^*)$ and AddTension$\overleftarrow{RL}(-2\Delta, f_r^*, f_\ell^*)$.

Now we must actually update the shortest path tree and the cotree. Let z by the child of y in T. We call Cut(zy) and then Join(xy) to connect the root $s$to y via the shorter path. This also (conceptually) recolors y and its subtree blue. To update the dual tree C, we call $cut(xy)$ in the dual structure to remove the edge xy whose tension is no longer changing, and then Join$(z, y, 0, -w(zy))$ to insert zy, which has $t(\overrightarrow{zy}) = 0$ and $t(\overrightarrow{yz}) = -w(zy)$; this reconnects our cotree to match our new shortest path tree.

Each edge change in the shortest path tree calls a constant number of operations, taking $O(\log n)$ amortized time total. We next bound the number of edges that must be added or removed from the graph.

Lemma 3.3.1. *Shortest path trees in planar graphs can be represented in such a way that the shortest path tree T_u rooted at u can be changed to the shortest path tree T_v rooted at a neighbor v of u in $O(k \log n)$ time, where k is the number of edges of T_u not present in T_u. In this representation, any shortest path distance from the root can be computed in $O(\log n)$ time.* □

In the next section, we show that any edge can be inserted or deleted a constant number of times.

3.3.2 Analysis for Planar Graphs

Klein [82] noted that if one maintains the so-called *leftmost shortest path tree*, each edge enters and leaves the shortest path tree a constant number of times. This property also holds for our algorithm.

Lemma 3.3.2. *As the root of the shortest path tree moves along f, each edge enters and leaves the shortest path tree $O(1)$ times.*

Proof: Recall that we may assume shortest paths are unique; this can be enforced using standard perturbation techniques [97].

A green edge xy enters the shortest path tree when either $t(\overrightarrow{xy}) = 0$ or $t(\overrightarrow{yx}) = 0$. We restrict our attention to the case $t(\overrightarrow{xy}) = 0$, since the other $t(\overrightarrow{yx}) = 0$ is symmetric.

Let A be the set of points on δf whose shortest path trees contain $\overrightarrow{xy}$. The vertices in A are the the roots of shortest path trees that could contain $\overrightarrow{xy}$. Let $B = \delta f \setminus A$, the vertices of f not contained in A.

Suppose A is disconnected. We can then find two shortest paths p and q from vertices in A to the vertex y which use edge $\overrightarrow{xy}$. But then there must be a shortest path π from a vertex of B to y which crosses p or q, since p and q

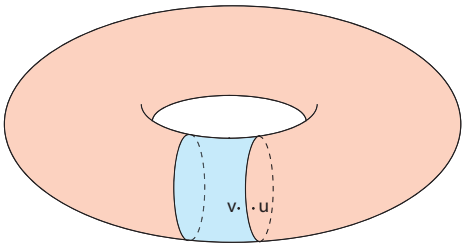

Figure 3.5. As the blue tree expands, the set of green edges could intersect itself and separate into $g + 1$ components. Here, the blue subtree (rooted at v) wrapped around the torus, and the set of green edges is shown as two disconnected cycles in the dual which separate the red subtree from the blue subtree.

together with the boundary of f form a closed curve. This is a contradiction; shortest paths cannot cross, because (by our earlier assumption) shortest paths are unique and any crossing would allow a shorter path. Thus, A is connected.

Now consider maintaining the shortest path tree as the root $s$moves along δf. The edges $\overrightarrow{xy}$ enters the shortest path tree at one end of A and leaves when $s$reaches the other end. Thus each (undirected) edge can be inserted or removed at most a total of 4 times.

□

Like Klein [82], we may assume the graph has bounded degree, which allows the use of persistence [47] to store and search any previous versions of the shortest path tree. The persistent data structure requires $O(n \log n)$ space. Thus:

Theorem 3.3.3 (Klein [82]). *Let G be a plane graph with n vertices, and let be f a face of G. After $O(n \log n)$ preprocessing time and using $O(n \log n)$ space, a shortest path distance from any vertex on f to any other vertex can be found in $O(\log n)$ time.*

3.4 Higher Genus Graphs

3.4.1 Algorithm for Orientable Surfaces

The algorithm in the planar case does not immediately extend to higher genus graphs because the dual graph $G^* \setminus E(T)^*$ is no longer a tree. Without this tree structure, the set of green edges can have a much more complicated combinatorial structure, and therefore finding tense edges is more difficult. For example, suppose $g = 1$. Initially, the blue subtree is divided from the red subtree by a cycle of green edges. However, as the blue subtree grows, it is possible for the blue to meet at other places, and the green boundary splits into two connected components. See Figure 3.5.

Without loss of generality, we assume each vertex has degree at most 3. Let T be a shortest path tree rooted at a point $s$along the edge uv, and assume that uv is in T. We decompose the dual edges $C = ((G \setminus T) \cup \{uv\})^*$ as follows.

Iteratively delete all edges of degree 1. These edges form a forest in the dual graph, which we denote F. After the edges of F have been deleted, we have a set of paths, which we call *cut paths*, meeting at vertices of degree 3; this is the *cut locus* for the root of the shortest path tree [59]. Denote these cut paths as $P = \{\pi_0, \pi_1, \ldots, \pi_k\}$, where each π_i is a path between two dual vertices of degree 3, and π_0 denotes the path containing e^*. (This is the *reduced cut locus* described in [58] and [59].) Cutting the surface along the paths $P \setminus \{\pi_0\}$ gives a topological disk, and Euler's formula implies that $k = O(g)$ [58, Lemma 4.2]. [1] Since π_0 has its endpoints in the boundary of the disk, cutting the surface along the paths splits the surface into two topological disks, R and B.

Consider now sliding the source $s$of the shortest path tree T across uv. See Figure 3.6. Let disk R contain all the red vertices, and disk B contain all the blue vertices. The set of green edges, or primal edges whose endpoints are not monochromatic, are again those edges that could potentially become tense. The green edges cross the boundary between B and R. Therefore, P contains the dual of every green edge. Moreover, for any path $\pi_i \in P$, either all of the edges in $(\pi_i)^*$ are green or none of them are. We say that a cut path is *green* when all its dual edges are green.

We maintain the primal tree T in an Eulertour tree exactly as in the planar case.

For the dual structure, we partition the dual subgraph C into trees $C_0, C_1, \ldots$ by attaching each tree in F to an adjacent path π_i; each tree C_i is stored in a self-adjusting top tree. The trees hanging from a vertex where cut paths meet are assigned to and stored in only one of the dual trees C_i. Recall that self-adjusting top trees are stored using a path decomposition; here, π_i is stored as the root path of C_i (using the EXPOSE operation).

We also maintain a representation of the reduced cut locus Φ, and embedded graph whose edges correspond to the cut paths in P. The graph has $O(g)$ edges and two faces, R and B. To determine if a cut path π_i is green, we simply find the corresponding edge in Φ and check if it bounds both R and B in $O(g)$ time.

Consider a green cut path π_i, with endpoints f_i^* and g_i^*. We can find in $O(\log n)$ amortized time a directed edge $\overrightarrow{xy}$ with maximum tension among those crossing π_i as follows. Assume without loss of generality that $x \in B$ and $y \in R$. The path π_i is $\pi(C_i, f_i^*, g_i^*)$ (without orientation). We can determine if the oriented path $\pi(C_i, f_i^*, g_i^*)$ bounds the blue disk B to its right or to its left by looking in Φ. If $\pi(C_i, f_i^*, g_i^*)$ bounds B to its left, MAXTENSION$\overrightarrow{LR}(f_i^*, g_i^*)$ returns the edge with maximum tensions crossing π_i; otherwise, MAXTENSION$\overleftarrow{RL}(f_i^*, g_i^*)$ returns the edge with maximum tension. A similar argument shows that we can update the tensions for all arcs crossing any green cut path in $O(\log n)$ amortized time.

Our algorithm begins with a shortest path tree T rooted at a vertex $s= u$.

[1] The concept of cut path in [58] is different, in that it does not include π_0. This "extra cut path" in our definition may produce up to three extra paths in our setting.

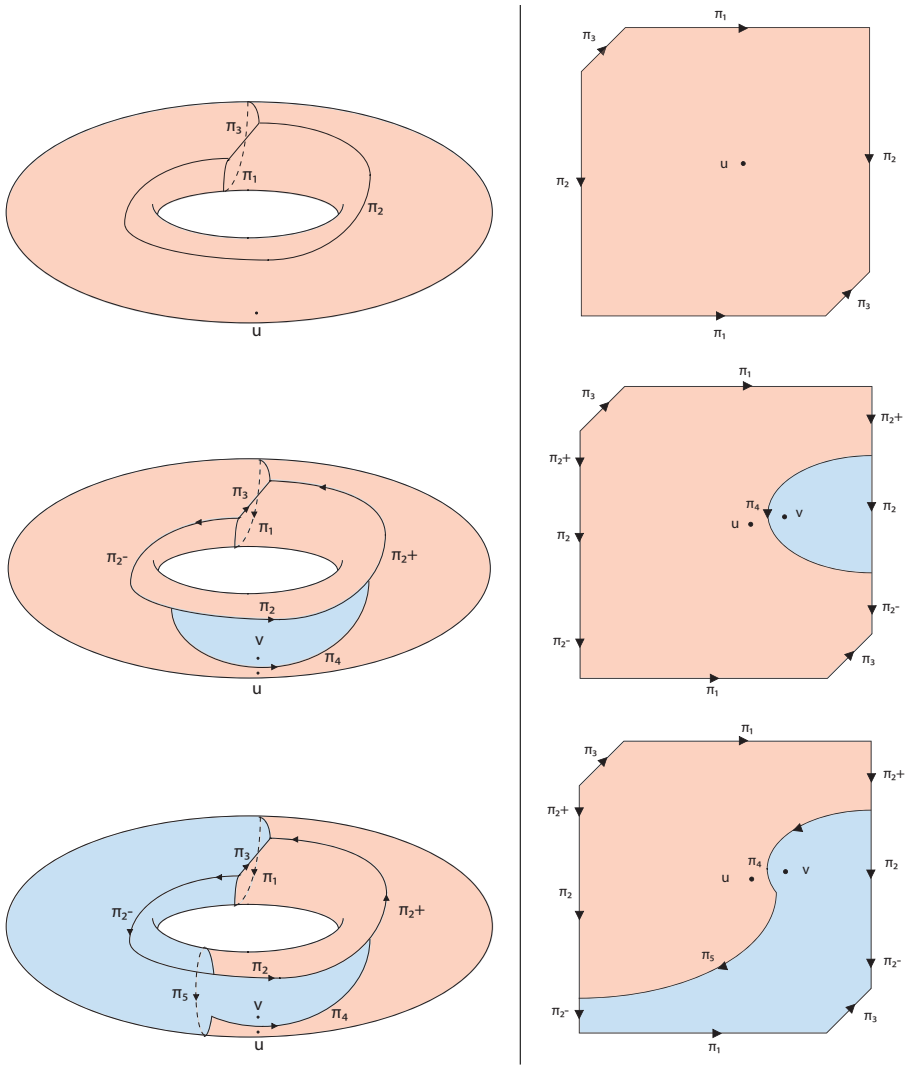

Figure 3.6. An example of the algorithm progressing. On the left, the alterations are shown on the surface on the torus, and on the right, they are shown on a polygonal schema. As new edges become tense, the set of cut paths alters, but always separates the blue subtree from the red subtree.

If the edge uv is not in the shortest path tree, all vertices are red. When uv becomes tense, it enters the shortest path tree, and the subtree rooted at v becomes blue.

Once $uv \in T$, the algorithm repeats the following steps until $s$reaches v or uv leaves the shortest path tree (meaning that every vertex is blue). We find the first directed edge $\overrightarrow{xy}$ that will become tense by querying every green cut path for its tensest dual edge. We take $O(\log n)$ amortized time per cut path, and there are $O(g)$ paths, so it takes $O(g \log n)$ amortized time to find the edge $\overrightarrow{xy}$ with maximum tension. Let $\Delta = t(\overrightarrow{xy})/2$; this is the distance $s$must go along uv for the edge $\overrightarrow{xy}$ to enter the shortest path tree.

Next, we update the shortest path tree to simulate moving the source sa distance Δ along the uv. Let z be the child of y in the shortest path tree. We

then call AddSubtree(u, δ) and AddSubtree$(v, -\delta)$ to update the distances, and Cut(zy), and Join(x, y) to update the structure of the tree.

Next, we update the tensions in the dual structure. For each green cut path, we add $+2\Delta$ to every arc from blue to red and add -2Δ to every arc from red to blue.

We also update Φ and the trees C_i to reflect the new cut paths. Edge yz is now green, and an endpoint of $(yz)^*$ can be vertices in F or directly on some cut path π_j. Find the unique (and possibly empty) path in F connecting each endpoint of $(zy)^*$ to the paths P, by performing an expose in the self-adjusting top tree which each endpoint belongs to; call these paths π_z and π_y. Then $\pi = \pi_z \circ (zy)^* \circ \pi_y$ is the new cut path we must add to C.

As in the planar case, we first call Cut$((xy)^*)$ and Join$(z, y, 0, -w(yz))$ to update the dual forest. Note that Join will also make π the top level path in the combined self-adjusting top tree by calling expose on that path. Now let pi_l be the cut path that $\overrightarrow{xy}$ crossed. π_l is no longer a cut path, so we find the cut paths in C which meet the two endpoints of π_l at vertices of degree three in Φ. The self-adjusting top trees left after the operation Cut$((xy)^*)$ will be joined to one of these other cut paths, since they are part of F.

We still need to adjust the partition of C so that the cut paths are correct. To do this, we call Junction to find the vertices a and b where π intersects the two other cut paths, say π_i and π_j, respectively. The portion of these two cut paths between a and $(xy)^*$ and b and $(xy)^*$ is no longer green, since the subtree rooted at y is now blue. We call expose on a and the relevant vertex of π_i in order to update the representation of π_i's self-adjusting top tree, and similarly update π_j.

Finally, we update Φ by removing the edge corresponding to π_l and adding the new edge corresponding to π. Since Φ has size $O(g)$, this takes $O(g)$ time.

This finishes the description of the algorithm to move $s$along a single edge uv. We handle each pivot, or change in the shortest path tree, in $O(g \log n)$ amortized time. The number of pivots is bounded by the difference between T_u and T_v. This gives the following:

Lemma 3.4.1. *Given a shortest path tree rooted at a vertex v in a graph that is embedded on an orientable surface of genus g, we can compute the shortest path tree rooted at a neighbor u of v in $O(kg \log n)$ time, where k is the number of edges in T_u which are not in T_v. Shortest path distances in the tree can then be computed in $O(\log n)$ time.*

3.4.2 Analysis for Higher Genus Graphs

Let G be a graph embedded in a surface of genus g, orientable or not. We first show a bound on the number of times that an edge can enter or leave the shortest path tree as the root $s$moves around a face.

Lemma 3.4.2. *As the source of the shortest path tree moves along a face f, each edge enters or leaves the shortest path tree $O(g)$ times.*

Proof: We use an argument similar to Lemma 3.3.2. We bound the number of times any edge xy becomes tense and enters the shortest path tree. As before, let A be the set of points on the boundary of f such that $\overrightarrow{xy}$ is in T_s, where T_s is the shortest path tree rooted at s. A is the union of a set of disjoint, maximal paths $A_1, \ldots, A_k$ on the boundary of f. Edge $\overrightarrow{xy}$ enters the shortest path tree exactly k times, once at the initial endpoint of each A_i. We will argue that $k = O(g)$, which will conclude the proof.

Fix a point v_i in each component A_i, for $i = 1 \ldots k$. Next, let p_i be the shortest path from v_i to y. By construction, p_i uses the edge $\overrightarrow{xy}$, and for all $i \neq j$, paths p_i and p_j do not cross.

Let $\mathcal{N}$ be the surface obtained by contracting the face f to a point f. We claim that in $\mathcal{N}$, any two paths p_i and p_j with $i \neq j$ are non-homotopic. Assume for the purpose of contradiction that p_i and p_j are homotopic in $\mathcal{N}$. In $\mathcal{M}$, this means that there is a subwalk f' of f that together with p_i and p_j bound a disk (and thus a planar graph). But then the shortest path to y from every vertex of f' uses xy, which implies that v_i and v_j are in the same connected component of A, which is impossible.

Finally, [25, Lemma 2.1] implies that the number of pairwise non-crossing, non-homotopic paths in $\mathcal{N}$ is $O(g)$, since we can combine pairs of non-homotopic paths to get a set of pairwise non-crossing, non-homotopic loops with basepoint f^*. We conclude that $k = O(g)$, which completes the proof. □

Since each update to the tree takes $O(g \log n)$ amortized time and there are $O(gn)$ possible updates, the total running time of our algorithm is $O(g^2 n \log n)$. Again, we can use standard techniques to convert to a graph with bounded degree, and use persistence [47] to store and search any previous versions of the shortest path tree.

During our algorithm, we need $O(g + n) = O(n)$ space to maintain the (dual) structures. The primal structure also uses $O(n)$ space is updated $O(gn)$ times. Therefore, the final data structure for storing all shortest path trees uses $O(g^2 n \log n)$ space. We conclude:

Theorem 3.4.3. *Let G be a graph with n vertices embedded in a surface of genus g, and let be f a given face of G. With $O(g^2 n \log n)$ preprocessing time, a shortest path distance from any vertex on f to any other vertex can be found in $O(\log n)$ time.*

3.5 Computing Shortest Nonseparating and Noncontractible Cycles

In this section we describe algorithms to find the shortest nonseparating and shortest noncontractible cycles in a combinatorial surface $\mathcal{M}$, orientable or not. Our technique for maintaining shortest path trees is condensed in the following lemma.

Lemma 3.5.1. *Let α be a simple cycle or arc in $\mathcal{M}$. A shortest cycle crossing α exactly once can be obtained in $O(g^2 n \log n)$ time.*

Proof: Consider the surface obtained by cutting $\mathcal{M}$ along α: each vertex v in α gives rise to two vertices v' and v'', and two boundary arcs or cycles α' and α''. Let $\mathcal{N}$ be the surface obtained by gluing disks to the boundaries that contain α' and α''. (If α is an arc, then α' and α'' may be contained in a single boundary.) A cycle in $\mathcal{M}$ that crosses α once at a point v becomes a path in $\mathcal{N}$ connecting v' to v''. Thus, a shortest cycle that crosses α once at v is a shortest path that connects v' to v'' in $\mathcal{N}$, and vice versa. Since all the points v', with $v \in \alpha$, belong to a face of $\mathcal{N}$, we can find in $O(g^2 n \log n)$ time a closest pair (v'_0, v''_0) by Theorem 3.4.3. Computing the shortest path from v'_0 to v''_0 gives the desired path. □

For simple arc or cycle α, let $\mathit{Cross}(\alpha)$ be the set of cycles which cross α exactly once. If α is separating, then $\mathit{Cross}(\alpha) = \emptyset$ because every cycle crosses α an even number of times. We also note that every cycle in $\mathit{Cross}(\alpha)$ is noncontractible, because contractible cycles are also separating, and therefore any cycle or arc must cross it an even number of times.

3.5.1 Shortest Nonseparating Cycle

Our algorithm works both for orientable and nonorientable surfaces. Consider a surface $\mathcal{M}$. It suffices to consider surfaces without boundary, since any shortest nonseparating cycle in $\mathcal{M}$ is also nonseparating in the surface obtained by attaching disks to the boundaries.

Cabello and Mohar [19] describe how to construct in $O(gn \log n)$ time a set S of $O(g)$ simple cycles such that the shortest cycle in $\bigcup_{\ell \in S} \mathit{Cross}(\ell)$ is a shortest nonseparating cycle. Since S consists of $O(g)$ loops, we can apply the previous lemma to each $\mathit{Cross}(\ell)$ for each ℓ in S and take the globally shortest cycle. We conclude:

Theorem 3.5.2. *Let $\mathcal{M}$ be an orientable surface, possibly with boundary, of complexity n and genus g. We can find a shortest nonseparating cycle in $O(g^3 n \log n)$ time.*

3.5.2 Shortest Noncontractible Cycle

The main technique for noncontractible cycles is to find a curve which intersects a shortest noncontractible cycle at most once and whose removal decreases the genus or number of boundaries of the surface. Our main tool to prove that such a curve exists is the following exchange argument.

Lemma 3.5.3. *Let $\mathcal{M}$ be a surface and let ℓ_x be a shortest noncontractible loop with given basepoint $x \in \mathcal{M}$. There is a shortest noncontractible cycle in $\mathcal{M}$ crossing ℓ_x at most once.*

Proof: Let C be a shortest noncontractible cycle that crosses ℓ_x the minimum number of times. We claim that C crosses ℓ_x at most once. Assume for the purpose of contradiction that C crosses ℓ_x twice, at y and z. Let γ_1 and γ_2 be the two subpaths of C from y to z, let β_1 and β_2 be the subpaths of ℓ_x from y to z. To simplify notation, we do not differentiate between a path and its reverse. We consider two cases and in each case arrive to a contradiction with the hypothesis that C and ℓ_x cross twice.

First, suppose $\beta_1 \sim \gamma_1$; the other cases are symmetric. The loop $(\beta_2 + \gamma_1)$ passes through x, and it is noncontractible because $(\beta_2 + \gamma_1) \sim (\beta_2 + \beta_1) = \ell_x$. We then have $|\beta_2| \leq |\gamma_1|$ because $|\ell_x| \leq |\beta_1| + |\gamma_1|$. The cycle $\tilde{C} = \gamma_2 + \beta_2$ is noncontractible because $(\gamma_2 + \beta_2) \sim (\gamma_2 + \gamma_1) = C$. It also crosses ℓ_x at least twice less than C, and $|\tilde{C}| = |\gamma_2| + \beta_2| \leq |\gamma_2| + |\gamma_1| \leq |C|$. This contradicts the definition of C.

It remains to consider the case where no pair of paths β_i and γ_i are homotopic. The cycles $\beta_1 + \gamma_1$ and $\beta_1 + \gamma_2$ are noncontractible; otherwise, $\gamma_1 \sim \beta_2$ or $\gamma_2 \sim \beta_2$, which contradicts our assumption. Now $\beta_1 + \gamma_1$ is a noncontractible loop through x, and therefore $|\ell_x| \leq |\beta_1| + |\gamma_1|$, so $|\beta_2| \leq |\gamma_1|$. The cycle $\tilde{C} = \beta_2 + \gamma_2$ is noncontractible because $\beta_2 \nsim \gamma_2$. $\tilde{C}$ also crosses ℓ_x two fewer times than C, and $|\tilde{C}| = |\beta_2| + |\gamma_2| \leq |\gamma_1| + \gamma_2| = |C|$, which contradicts our choice of C. □

The following lemma discusses what happens with simple noncontractible cycles when pasting a disk into a boundary of the surface.

Lemma 3.5.4. *Let $\mathcal{M}$ be a surface with boundary and let δ be one of its boundary components. Let $\mathcal{N}$ be the surface obtained by pasting a disk to δ. A noncontractible simple cycle in $\mathcal{M}$ is either noncontractible in $\mathcal{N}$ or homotopic to δ in $\mathcal{M}$.*

Proof: Consider a noncontractible simple cycle C in $\mathcal{M}$. Let D_δ be the disk that is attached to $\mathcal{M}$ to obtain $\mathcal{N}$. If C is contractible in $\mathcal{N}$, then C bounds a disk D_C in $\mathcal{N}$. Note that the disk D_C must contain D_δ; otherwise, C would also bound a disk in $\mathcal{M}$, implying that C is contractible in $\mathcal{M}$. $D_C \setminus D\delta$ is an annulus in $\mathcal{M}$ with boundary cycles C and δ. It follows that C and δ are homotopic in $\mathcal{M}$. □

We also have the following results regarding arcs in a surface with boundary.

Lemma 3.5.5. *Let $\mathcal{M}$ be a surface with boundary, let δ be one of its boundary cycles, and let α be a shortest noncontractible arc with endpoints in δ. There is a shortest noncontractible cycle in $\mathcal{M}$ that is either homotopic to δ or that crosses α at most once.*

Proof: Let C be a shortest noncontractible cycle in $\mathcal{M}$. Assume $C \nsim \delta$, since otherwise the proof is complete. Lemma 3.5.4 implies that C is a shortest noncontractible cycle in $\mathcal{N}$, the surface obtained by pasting a disk D at δ. In the surface $\mathcal{N}$, place a new vertex p in D and connect it through edges with weight L to each vertex of δ. Consider the loop ℓ with basepoint p that follows the edge $p\,\alpha(0)$, the arc α, and the edge $\alpha(1)\,p$. If we choose L large enough, ℓ is a shortest noncontractible loop in $\mathcal{N}$ through p, so Lemma 3.5.3(a) implies that a shortest noncontractible cycle C' in $\mathcal{N}$ crosses ℓ at most once. We must have$|C'| = |C|$, and if L is large enough, C' must avoid the disk D. It follows that C' is a shortest noncontractible cycle in $\mathcal{M}$ that crosses α at most once. □

Lemma 3.5.6. *Let $\mathcal{M}$ be a surface with at least two boundary components, and let α be a shortest arc connecting two different boundaries of $\mathcal{M}$. There is a shortest noncontractible cycle in $\mathcal{M}$ that crosses α at most once.*

Proof: Let C be a shortest noncontractible cycle in $\mathcal{M}$ that crosses α the minimum number of times. We claim that C crosses α at most once. Assume for the purpose of contradiction that C crosses α at least twice, and let y and z be two such crossings. Let γ_1 and γ_2 be the two subpaths of C from y to z, let $\tilde{\alpha}$ be the subpath of α from y to z. Since α is a shortest arc connecting the two specified boundaries, we have $|\tilde{\alpha}| \leq |\gamma_1$ and $|\tilde{\alpha}| \leq |\gamma_2|$. If $\tilde{\alpha} \nsim \gamma_1$, then we have a contradiction: the cycle $\tilde{\alpha} + \gamma_1$ is noncontractible, crosses α fewer times than C does, and $|\tilde{\alpha}| + |\gamma_1| \leq |\gamma_2| + |\gamma_1| = |C|$. Similarly, we must have $\tilde{\alpha} \sim \gamma_2$. But then $\gamma_1 \sim \gamma_2$, which implies that C is contractible, which is a contradiction. □

The next lemma summarizes the algorithmic tools that we will use.

Lemma 3.5.7. *Let $\mathcal{M}$ be a surface of complexity n.*

(a) *Given a basepoint x, we can find in $O(n \log n)$ time a shortest noncontractible loop with basepoint x. This loop has multiplicity 2.*

(b) *Given a boundary δ, we can find in $O(n \log n)$ time a shortest cycle homotopic to δ.*

(c) *Given a boundary δ, we can find in $O(n \log n)$ time a shortest noncontractible arc with endpoints in δ. This acr has multiplicity at most 2 and is edge-disjoint from δ.*

(d) *If $\mathcal{M}$ has two or more boundaries, we can find in $O(n \log n)$ time a shortest arc with endpoints in two specified boundaries. This arc has multiplicity 1 and is edge-disjoint from all boundaries of $\mathcal{M}$.*

Proof: (a) See Erickson and Har-Peled [58, Lemma 5.2].

(b) See Cabello et al [17].

(c) Contract δ to a point p and construct a shortest noncontractible loop with basepoint p as in item (i). This is the desired arc in $\mathcal{M}$.

(d) Select boundaries δ and δ' of $\mathcal{M}$, contract them to points p and p', and construct in $O(n \log n)$ time the shortest path from p to p'.

□

Lemma 3.5.8. *Let $\mathcal{M}$ be a combinatorial surface with complexity n, genus g, and b boundaries. Let $\mathcal{N}$ be the surface obtained by pasting a disk to each boundary of $\mathcal{M}$. We can find a shortest noncontractible cycle in $\mathcal{M}$ in $O(bn \log n)$ time plus the time used to find a shortest noncontractible cycle in $\mathcal{N}$.*

Proof: Number the boundaries $\delta_1, \delta_2, \ldots, \delta_b$, and set $\mathcal{M}_0 = \mathcal{M}$. For $i \geq 1$, let $\mathcal{M}_i$ be the surface obtained from $\mathcal{M}_{i-1}$ by pasting a disk to δ_i. In particular, $\mathcal{M}_b = \mathcal{N}$. For each i, let C_i be a shortest cycle homotopic to δ_i in $\mathcal{M}_{i-1}$. We can compute each cycle C_i in $O(n \log n)$ time.

Lemma 3.5.4 implies that a shortest noncontractible cycle in $\mathcal{M}_{i-1}$ is either C_i or a shortest noncontractible cycle in $\mathcal{M}_{i-1}$. Thus, the shortest noncontractible cycle in $\mathcal{M}$ is either the shortest among $C_1, \ldots, C_b$ or a shortest noncontractible cycle in $\mathcal{N}$. □

Theorem 3.5.9. *Let $\mathcal{M}$ be a combinatorial surface with complexity n, genus g, and b boundaries. We can find a shortest noncontractible cycle of $\mathcal{M}$ in $O((g^3 + b)n \log n)$ time.*

Proof: We apply Lemma 3.5.8 to $\mathcal{M}$. We spend $O(bn \log n)$ time, and have reduced the problem to find a shortest noncontractible cycle in a surface $\mathcal{N}$ of genus g and no boundary. We next give an iterative algorithm that reduces either the genus or the number of boundaries of the subproblems in each iteration. The algorithm stops when each remaining component is a topological disk. We distinguish three cases. $\mathcal{N}$ denotes the surface in the subproblem; note that $\mathcal{N}$ has complexity $O(n)$ and genus at most g.

(a) If $\mathcal{N}$ is a surface without boundary, we choose a point $x \in \mathcal{N}$ and find a shortest noncontractible loop ℓ_x through x. Lemma 3.5.3 implies that there is a shortest noncontractible cycle in $\mathcal{N}$ crossing ℓ_x at most once. We can compute the shortest cycle in $\mathit{Cross}(\ell_x)$ and then set $\mathcal{N} = \mathcal{N} \setminus \ell_x$. Note that if ℓ_x is separating, then $\mathcal{N}'$ has two connected components

and $Cross(\ell_x) = \emptyset$. Lemmas 3.5.1 and 3.5.7(a) imply that we spend $O(g^2 n \log n)$ time in this iteration.

(b) If $\mathcal{N}$ has exactly one boundary δ, we find a shortest noncontractible arc α with endpoints in δ. Lemma 3.5.5 implies that there is a shortest noncontractible cycle in $\mathcal{N}$ that either crosses α at most once or is homotopic to δ. We can compute the shortest cycle in $Cross(\ell_x)$ and teh shortest cycle homotopic to δ, set $\mathcal{N} = \mathcal{N} \setminus \ell_x$. Lemmas 3.5.1 and 3.5.7(b-c) imply that we spend $O(g^2 n \log n)$ time in this iteration.

(c) If $\mathcal{N}$ has two or more boundaries, we find a shortest arc α connecting them. Lemma 3.5.6 implies that there is a shortest noncontractible cycle in $\mathcal{N}$ crossing α at most once. We compute the shortest cycle in $Cross(\ell_x)$ and replace $\mathcal{N}$ with $\mathcal{N} \setminus \ell_x$. Lemmas 3.5.1 and 3.5.7(d) imply that we spend $O(g^2 n \log n)$ time in this iteration.

This finishes the description of the algorithm. The algorithm has $O(g)$ iterations; starting with no boundary, we iteration in case (a) once and then cases (b) or (c) at most g times. The arcs and loops that we cute, which are obtained from Lemma 3.5.7, have multiplicity at most two and are edge-disjoint from the boundary. Thus any subsurface $\mathcal{N}$ considered in a subproblem has at most four copies of an edge of $\mathcal{M}$. Thus, $m = O(n)$, and in each iteration we spend $O(g^2 n \log n)$ time. □

Chapter 4

Shortest Splitting Cycles

Another natural question on surfaces is to find simple, noncontractible, and separating cycles, or *splitting* cycles. In the combinatorial surface model, this question is particularly nice in that it serves as a possible generalization of planar graph separators [88], since it separates both the graph and the topology.

In this chapter, we prove in Section 4.2 that computing the shortest splitting cycle on a given surface is NP-hard. In Section 4.3, we prove that a shortest splitting cycle on a surface of genus g with b boundary components cuts any shortest path on the surface $O(g+b)$ times; this bound is tight if the surface has no boundary. This property leads to an algorithm to compute a shortest splitting cycle in $(g+b)^{O(g+b)}n \log n$ time, which we describe in Section 4.4. Thus, we show that the shortest splitting problem is fixed-parameter tractable with respect to the genus and the number of boundary components of the surface. This is the first result of this kind among the previously cited works. In particular, although Erickson and Har-Peled provide an algorithm to compute the minimum cut graph on any surface of constant genus and constant number of boundary components in polynomial time, the order of the polynomial depends on the genus and on the number of boundary components [58].

In Section 4.5, we consider the problem of finding a shortest cycle that splits the surface into two surfaces of prescribed topology. For example, given a surface of genus g with b boundary components, and given $g' \leq g$ and $b' \leq b+1$, we can find the shortest cycle that splits $\mathcal{M}$ into two surfaces, one of which has genus g' and b' boundary components. We can also find the shortest splitting cycle that is not homotopic to a boundary. We give an algorithm to solve these questions in $(g+b)^{O(g+b)}n \log n$ time.

Finally, iteratively cutting a surface of genus g without boundary along a shortest splitting cycle decomposes the surface into g punctured tori, but we show in Section 4.6 that one does not necessarily obtain the shortest such decomposition with this method.

We emphasize that most of our structural results (Lemma 4.3.1, Proposition 4.3.2, and Theorem 4.3.3) are *not* limited to the combinatorial surface model, but rather apply to surfaces with a wide range of metrics, including piecewise-linear, piecewise-algebraic, and abstract Riemannian surfaces. Even the NP-hardness reduction in Section 3 can be easily modified to apply in these

more general surface models. The restriction to the combinatorial surface model is only necessary for our algorithmic results in Section 4.4, largely because this is the only known surface model in which exact shortest paths can be computed efficiently without additional assumptions.[1]

4.1 Preliminaries

4.1.1 Preliminary Lemma

Let L be a set of simple, pairwise disjoint loops with common basepoint, all in the interior of $\mathcal{M}$. Let $\mathcal{M}'$ be a connected component of $\mathcal{M} \setminus L$. If $\mathcal{M}'$ is a disk with one, two, or three copies of the basepoint on its boundary, it is called respectively a *monogon*, a *bigon*, or a *trigon*. If i$\mathcal{M}'$ is an annulus with one copy of the basepoint on its boundary and a boundary cycle that is a boundary of the original surface $\mathcal{M}$, then $\mathcal{M}'$ is called an *elementary annulus*.

Lemma 4.1.1. *Let $\mathcal{M}$ be a surface with genus g and b boundaries. Let $L \neq \emptyset$ be a set of simple, pairwise disjoint loops with common basepoint, all in the interior of $\mathcal{M}$. If no component of $\mathcal{M} \setminus L$ is a monogon or a bigon, then $|L| \leq 6g + 2b - 3$.*

Proof: We first prove that we can extend L to a set L' of simple, pairwise disjoint loops with the same basepoint, x, such that $\mathcal{M} \setminus L$ consists of trigons and elementary annuli only. Initially, let $L' = L$.

Let $\mathcal{M}'$ be any component of $\mathcal{M} \setminus L$; this component has at least one copy of x on its boundary. If $\mathcal{M}'$ contains a boundary component of $\mathcal{M}$, we add to L' a loop in $\mathcal{M}'$ that encloses this boundary component, splitting $\mathcal{M}'$ into an elementary annulus and another surface. Now we consider any component $\mathcal{M}'$ of $\mathcal{M} \setminus L'$ that is not an elementary annulus; such a component contains no boundary component of the original surface $\mathcal{M}$.

If $\mathcal{M}'$ has positive genus, we add to L' a nonseparating loop in $\mathcal{M}'$, based at some copy of x on its boundary. This does not create monogons or bigons and decreases the genus of $\mathcal{M}'$ by one. We repeatedly add nonseparating loops until $\mathcal{M}'$ has genus zero. If $\mathcal{M}'$ has at least two boundaries (and is not an elementary annulus), then each boundary contains a copy of the basepoint x, and we can add a path in $\mathcal{M}'$ that connects two copies of x on different boundaries. This path does not separate $\mathcal{M}'$, creates no monogon or bigon, and decreases the number of boundary components by one. So we may assume that $\mathcal{M}'$ is a disk.

[1] Efficient shortest-path algorithms for piecewise-linear surfaces [29, 93] require exact real arithmetic. Even if the input coordinates are integers, shortest path lengths are sums of square roots of integers; it is an open question whether two such sums can be compared in polynomial time on an integer RAM [12]. The analysis of these algorithms also assumes that the shortest path between any two points in the same face of the surface is contained in that face. Although this condition is satisfied by (even non-convex) polyhedra in any Euclidean space, it is not true for arbitrary abstract PL surfaces.

For some $k \geq 3$, the boundary of the disk $\mathcal{M}'$ contains k copies of x. Such a disk may be triangulated by adding $k-2$ loops. So we obtain a set $L' \supseteq L$ of simple, pairwise disjoint loops with common basepoint that split $\mathcal{M}$ into b elementary annuli and t trigons, for some integer t.

Finally, counting the edge-face incidences in two different ways, we obtain $2|L'| = 3t + b$. Euler's formula, applied to the surface $\mathcal{M}$ with each boundary component filled with a disk, implies $2-2g = 1-|L'|+t+b$. We conclude that $|L'| = 6g+2b-3$ and $t = 4g+b-2$. Since $|L| \leq |L'|$, the proof is complete. □

4.1.2 Finding a Splitting Cycle

If length is not a consideration, we can compute a splitting cycle on a surface $\mathcal{M}$ in $O(n)$ time as follows.

If $\mathcal{M}$ is a sphere, a disk, or a torus, then no splitting cycle exists. Otherwise, if $\mathcal{M}$ has at least one boundary component, then any cycle enclosing one boundary component is splitting. So we may assume that $\mathcal{M}$ has genus at least two and has no boundary.

First we construct a simple noncontractible cycle, using a variant of an algorithm of Erickson and Har-Peled [58], which finds the shortest noncontractible loop with a given basepoint in $O(n \log n)$ time. The running time is dominated by the computation of a shortest-path tree rooted at x using Dijkstra's algorithm. If we ignore the edge lengths and use breadth-first search instead, the running time drops to $O(n)$; the modified algorithm still returns a simple noncontractible cycle α. If α is separating, then we are done. Otherwise, we compute another noncontractible cycle that crosses α exactly once, in $O(n)$ time. To do this, we choose an arbitrary vertex $x \in \alpha$ and compute a simple path β from one copy of x to the other copy in $\mathcal{M} \setminus \alpha$, using (for example) depth-first search. The cycle $\gamma = \alpha \cdot \beta \cdot \bar{\alpha} \cdot \bar{\beta}$ is simple and null-homologous, but not contractible; specifically, one of the components of $\mathcal{M} \setminus \gamma$ is a punctured torus. Thus γ is a splitting cycle.

4.2 NP-Hardness

Theorem 4.2.1. *Finding the shortest splitting cycle on a combinatorial surface is NP-hard.*

Proof: A *grid graph of size n* is a graph induced by a set of n points on the two-dimensional integer grid. We describe a two-step reduction from the Hamiltonian cycle problem in grid graphs [80].

Let H be a grid graph of size n. To begin the first reduction, we overlay n 4×4 square grids of width $\epsilon < 1/4n$, one centered on each vertex of H. In each small grid, we color the square in the second row and second column *red* and the square in the third row and third column *blue* (where we fix the origin at

the upper left corner). We now easily observe that the following question is NP-complete: *Does the modified grid contain a cycle of length at most $n + 1/2$ that separates the red squares from the blue squares?* Any Hamiltonian cycle for H can be modified to produce a separating cycle of length at most $n+1/2$ by locally modifying the Hamiltonian cycle within each small grid, as shown in Figure 4.1, left and middle. Conversely, any separating cycle must pass through the center points of all n small grids, which implies that any separating cycle of length at most $n + 1/2$ must contain n grid edges that comprise a Hamiltonian cycle for H. We note that this result holds also if the cycle is allowed to visit vertices and edges of the modified grid several times, as in the case of combinatorial surfaces.

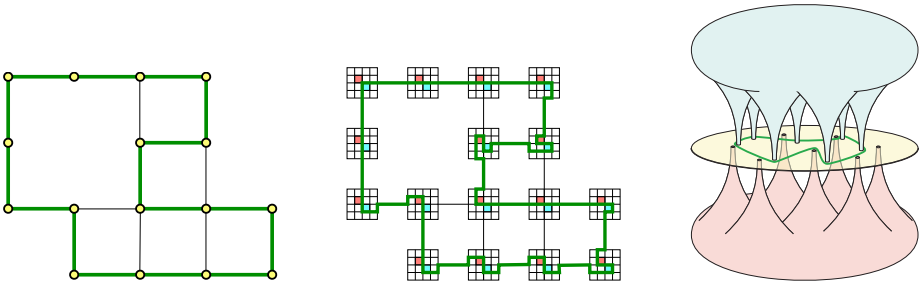

Figure 4.1. Left: A Hamiltonian cycle of length n. Middle: The corresponding red/blue separating cycle (not to scale). Right: Separating heaven from hell (not to scale); the central disk is a small portion of Earth.

In the second reduction, we reduce the problem to finding a minimum-length splitting cycle. We isometrically embed the modified grid on a sphere, which we call *Earth*. We remove the red and blue squares to create $2n$ punctures, which we attach to two new punctured spheres, called *Heaven* and *Hell*. We attach the n punctures in Heaven to the n blue punctures on Earth; similarly, we attach the n punctures in Hell to the n red punctures on Earth. We append edges of length $n + 1$ to the resulting surface so that each face of the final embedded graph is a disk. The resulting combinatorial surface $\mathcal{M}(H)$ has no boundary, genus $2n - 2$, and complexity $O(n)$; it can clearly be constructed in polynomial time. See Figure 4.1, right.

If the shortest cycle γ that splits $\mathcal{M}(H)$ has length less than $n+1/2$, then it must lie entirely on Earth. Moreover, γ must separate the blue punctures from the red punctures; otherwise, $\mathcal{M}(H) \setminus \gamma$ would be connected by a path through heaven or through hell. Thus, γ is precisely the shortest cycle that separates the red and blue squares in our intermediate problem. Testing whether γ has length less than $n + 1/2$ is thus NP-hard from the first reduction. □

With a few trivial modifications, our reduction also implies that computing the shortest splitting cycle on a polyhedral or Riemannian surface is NP-hard, although it is an open question whether the corresponding decision problems are still in NP.

4.3 Structural Properties

4.3.1 Multiplicity Bound

For any two points x and y on a cycle α, we let $\alpha[x, y]$ denote the path from x to y along α, taking into account the orientation of α. For a path or a dual edge α, the same notation is used for the unique simple path between x and y on α.

Lemma 4.3.1. *Any shortest splitting cycle on an orientable cross-metric surface $\mathcal{M}$ crosses each edge e^* of $G^*(\mathcal{M})$ at most once in each direction.*

Proof: Assume for the purpose of contradiction that some shortest splitting cycle γ crosses some dual edge e^* twice in the same direction, say left to right. Let x and z be consecutive left-to-right intersection points along e^*; that is, γ does not cross $e^*[x, z]$ from left to right. Then γ must cross e^* (exactly once) from right to left at some point y between x and z. Indeed, because γ is separating and $\mathcal{M}$ is orientable, the orientation of the crossings of e^* with γ must alternate along e^*.

The cycles $\gamma[x, y] \cdot e^*[y, x]$ and $\gamma[y, z] \cdot e^*[z, y]$ are noncontractible; otherwise, we could shorten γ by removing two crossings with e^* without changing its homotopy class.

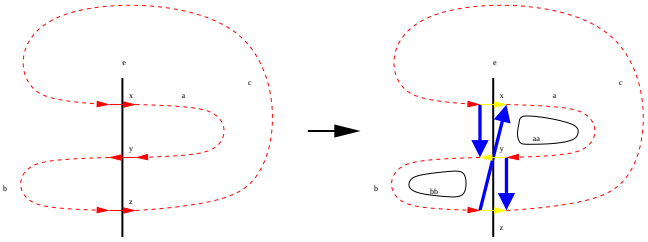

Figure 4.2. Lemma 4.3.1. If γ crosses e^* twice in the same direction, we can remove two crossings.

Define a new cycle $\gamma' = \gamma[x, y] \cdot e^*[y, z] \cdot \gamma[z, x] \cdot e^*[x, y] \cdot \gamma[y, z] \cdot e^*[z, x]$, as shown in Figure 4.2. The new cycle γ' is simple, because x, y, z are consecutive along e^*. The cycle γ' is in the same homology class as γ and is therefore null-homologous. By translating the noncontractible cycles $\gamma[x, y] \cdot e^*[y, x]$ and $\gamma[y, z] \cdot e^*[z, y]$ away from γ', we obtain two noncontractible cycles a' and b', one in each component of $\mathcal{M} \setminus \gamma'$; so γ' is noncontractible. Finally, γ' crosses e^* two times fewer than γ and crosses every other edge in $G^*(\mathcal{M})$ the same number of times as γ. We conclude that γ' is a splitting cycle that is shorter than γ, which is impossible. This concludes the proof.

As a side note, let $\mathcal{M}_\ell$ and $\mathcal{M}_r$ be the components of $\mathcal{M} \setminus \gamma$ on the left and on the right of γ; define $\mathcal{M}'_\ell$ and $\mathcal{M}'_r$ similarly to be the components of $\mathcal{M} \setminus \gamma'$ on the left and on the right of γ'. We remark that the topology of $\mathcal{M}_\ell$ and $\mathcal{M}'_\ell$ is the same, and similarly for $\mathcal{M}_r$ and $\mathcal{M}'_r$: topologically, $\mathcal{M}'_\ell$ is obtained from $\mathcal{M}_\ell$ by splitting it into two pieces along an arc, and by regluing these two pieces

along a subpath of their boundaries. A more formal way to see this result is to note that the Euler characteristic and the number of boundaries of $\mathcal{M}_\ell$ and $\mathcal{M}'_\ell$ are the same. □

4.3.2 Shortest-Path Crossing Bound

Proposition 4.3.2. *Let $\mathcal{M}$ be an orientable cross-metric surface with genus g and b boundary components. Let P be a set of shortest paths on $\mathcal{M}$ such that the intersection of any two shortest paths is a (possibly empty) set of common endpoints. There is a shortest splitting cycle that crosses each path in P at most $12g + 4b - 6$ times.*

Proof: Amongst all shortest splitting cycles, let γ have the minimum number of crossings with paths in P. We can assume that γ does not pass through the endpoints of any path in P, since we are on a cross-metric surface and could simply perturb γ slightly. Consider any path p in P that intersects γ.

The intersection points $\gamma \cap p$ partition γ into *arcs*. These arcs may intersect other paths in P. Let $\mathcal{M}/p$ be the quotient surface obtained by contracting p to a point p/p. The set of all arcs corresponds in $\mathcal{M}/p$ to a set of simple, pairwise disjoint loops L with basepoint p/p.

We claim that no component of $(\mathcal{M}/p) \setminus L$ can be a monogon. Otherwise, there would be two intersection points x and y of γ and p such that $\gamma[x, y]$ and $p[y, x]$ bound a disk. But then we could obtain a splitting cycle $\gamma[y, x] \cdot p[x, y]$, no longer than γ, that has fewer crossings with the paths in P, a contradiction.

We prove below that each loop in L is (in $\mathcal{M}/p$) incident to at most one bigon of $(\mathcal{M}/p) \setminus L$. Repeatedly remove any loop in L incident to a bigon until no bigons remain. Since each loop is incident to at most one bigon, at most half of the loops have been removed; so, if L' denotes the new set of loops, we have $|L| \leq 2|L'|$. Furthermore, no component of $(\mathcal{M}/p) \setminus L'$ is a monogon or a bigon, hence, by Lemma 4.1.1, we have $|L'| \leq 6g + 2b - 3$. So $|L| \leq 12g + 4b - 6$, as claimed.

To complete the proof, it thus suffices to prove that no loop in L bounds two bigons in $(\mathcal{M}/p) \setminus L$. We assume for the purpose of contradiction that some loop bounds two bigons. Then there are three arcs $u = \gamma[a, z]$, $v = \gamma[y, b]$, and $w = \gamma[c, x]$ such that u/p and v/p comprise the boundary of a component of $\mathcal{M}/p$ that is a disk, and similarly for v/p and w/p; see Figure 4.3, left. Since γ is separating, for a fixed orientation of $\mathcal{M}$, the clockwise boundaries of these disks are $u/p \cdot v/p$ and $\bar{v}/p \cdot \bar{w}/p$. Gluing these disks along v/p, we see that there is an open disk D_p in $\mathcal{M}/p$ bounded by $u/p \cdot \bar{w}/p$ such that v/p is the only loop in L intersecting this disk.

Consider the minimal subpath p' of p that includes the endpoints a, c, z, and x of u and w. In particular, the surfaces $(\mathcal{M}/p) \setminus (u \cup w)$ and $(\mathcal{M}/p') \setminus (u \cup w)$ are homeomorphic. One component of $\mathcal{M} \setminus (p' \cup u \cup w)$ is an open disk D such that the portion of γ inside D is precisely arc v.

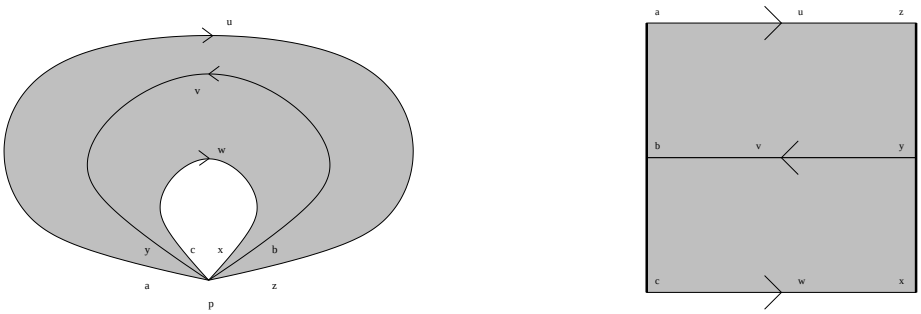

Figure 4.3. Left: Loop v/p is incident with two bigons. D_p is shaded. Right: A drawing obtained by expanding back the basepoint p/p. The two bold lines correspond to subsegments of p'. These two subsegments can overlap in the sense that a point on ac and a point on zx may correspond to the same point of p'.

The boundary of D is a walk in the embedded graph $p' \cup u \cup w$ and is composed of u, $\bar{w}$, and pieces of p'. Since the embedded graph $p' \cup u \cup w$ has no vertex of degree one, this boundary must in fact be $u \cdot p'[z, x] \cdot \bar{w} \cdot p'[c, a]$. See Figure 4.3, right.

We claim that p cannot enter the open disk D. Otherwise, a component q of $p \setminus p'$ would be entirely contained in D. Call s the common endpoint of q and p', where $s \in \{a, c, z, x\}$. By symmetry, we may assume $s = a$. Then, since w does not cross q and because the crossings are transverse, the arc u' preceding u along γ must be in D, whence $u' = v$. See Figure 4.4. It follows that $p[y, z]$ and $v \cdot u = \gamma[y, z]$ bound a disk. But γ does not cross $p[y, z]$, because otherwise there would be, in D, an arc different from v. We can thus replace the part $\gamma[y, z]$ of γ by $p[y, z]$. The resulting cycle γ' is simple, homotopic to γ, and no longer than γ; also, γ' crosses γ fewer times that γ. This contradicts the choice of γ.

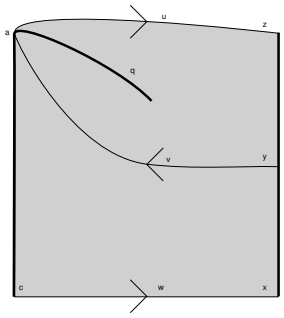

Figure 4.4. The two bold lines correspond to subsegments of p. Points s, y, and z must be pairwise distinct on $\mathcal{M}$ since γ is simple.

It follows that D is a component of $\mathcal{M} \setminus (p \cup u \cup w)$ with boundary $u \cdot p[z, x] \cdot \bar{w} \cdot p[c, a]$. There are two cases to consider depending on the relative directions of $p[z, x]$ and $p[c, a]$ along p.

If $p[z, x]$ and $p[c, a]$ are directed the same way along p, then $p[z, x]$ and $p[c, a]$ cannot strictly overlap since otherwise $\mathcal{M}$ would both be orientable and contain a Möbius strip. Up to a change of direction of γ and to an exchange between u

and w, we are in the situation depicted on top Figure 4.5. (The simplicity of γ implies that $c \neq x$.)

Suppose on the contrary that $p[z, x]$ and $p[c, a]$ have opposite directions along p. Up to a change of direction of γ and to an exchange between u and w, we can assume a arises first along p. We now claim that $p[z, x]$ and $p[c, a]$ cannot stricly overlap. Otherwise z would occur strictly between a and c, so that the arc u' following u along γ would enter D at z, whence $u' = v$. It follows that $z = y$ on $\mathcal{M}$ (refer to Figure 4.3, right). Again, since the crossings between γ and p are transverse, this implies that p enters D, contradicting a previous claim.

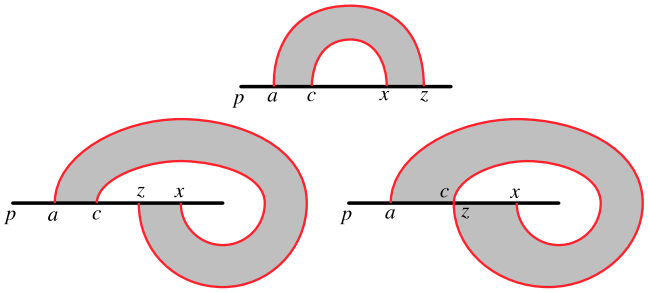

Figure 4.5. The three possible configurations for arcs $u = \gamma[a, z]$ and $w = \gamma[c, x]$; disk D is shaded.

We conclude the proof by showing that none of the configurations in Figure 4.5 can actually happen.

Top case: Recall that γ intersects D along arc v.

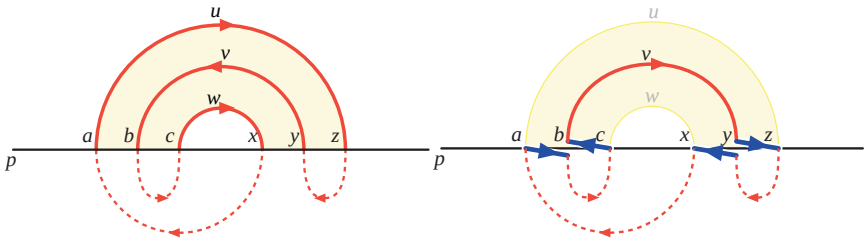

Figure 4.6. The exchange argument for the top configuration.

Without loss of generality, suppose the path $\gamma[x, a]$ does not contain any of the arcs u, v, or w. Consider the cycle

$$\gamma' = p[a, b] \cdot \gamma[b, c] \cdot p[c, b] \cdot \bar{\gamma}[b, y] \cdot p[y, z] \cdot \gamma[z, y] \cdot p[y, x] \cdot \gamma[x, a]$$

obtained by removing u and w from γ, reversing v, and connecting the remaining pieces of γ with subpaths of p; see Figure 4.6. This cycle crosses p fewer times than γ, and crosses any other path in p no more than γ. An argument identical to the proof of Lemma 4.3.1 implies that γ' is simple, null-homologous, and noncontractible. Because p is a shortest path, u cannot be shorter than $p[a, z]$, which implies that γ' is no longer than γ, contradicting our assumption that γ

is a shortest splitting cycle with the minimal number of crossings with paths in P.

Bottom left and right cases: As in the previous case, γ intersects D along v.

If $c = z$, there is only one way to connect these three arcs to form the cycle γ; if $c \neq z$, there are two possibilities. See Figure 4.7. In all three cases, by deleting arcs u and w, reversing arc v, and connecting the remaining pieces of γ with subpaths of p, we create a splitting cycle γ' that is no longer than γ and crosses the paths in P fewer times than γ, which is impossible. We omit the tedious details.

We note that, as in the proof of Lemma 4.3.1, the topology of the surface that is the part of $\mathcal{M}$ on either side of γ does not change during these exchanges: in all cases, topologically non-trivial components of this surface are cut along paths with endpoints on the boundary and reglued differently. □

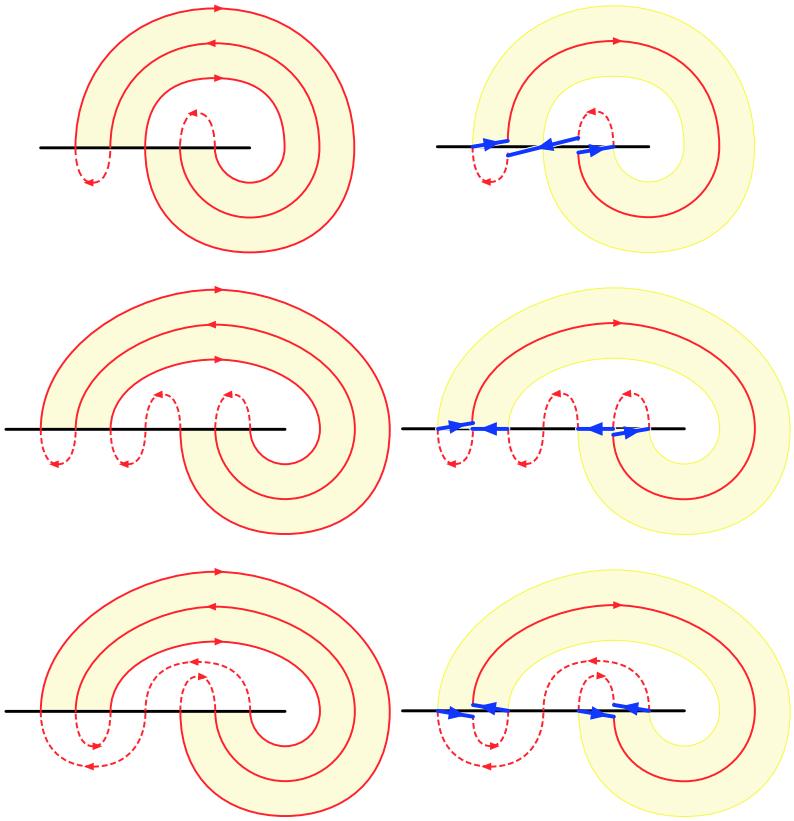

Figure 4.7. Three exchange arguments for the bottom configurations.

4.3.3 Crossing Lower Bound

Earlier algorithms for computing shortest noncontractible, nonseparating, or essential cycles rely on the fact that the desired cycle is the concatenation of

two equal-length shortest paths [95, 58].[2] Cabello and Mohar [19], Cabello [15], and Kutz [86] exploit a slightly different property to compute shortest nontrivial cycles more quickly on surfaces of constant genus: The desired shortest cycle crosses any shortest path at most twice. As we prove next, neither of these properties holds for the shortest splitting cycle; in particular, the upper bound of Proposition 4.3.2 is tight up to constant factors for surfaces without boundary.

Theorem 4.3.3. *For any $g \geq 2$, there is an orientable combinatorial surface $\mathcal{M}_g$ of genus g, without boundary, whose unique shortest splitting cycle (up to orientation) crosses a shortest path g times and (therefore) cannot be decomposed into fewer than g shortest paths and edges.*

Proof: We consider only the case where g is even. We construct a combinatorial surface $\mathcal{M}_g$ of genus g such that the shortest noncontractible cycle γ and the shortest splitting cycle μ cross g times; see Figure 4.8. The shortest noncontractible cycle γ can be partitioned into two shortest paths, one of which will be disjoint from μ; it follows that μ crosses some shortest path g times.

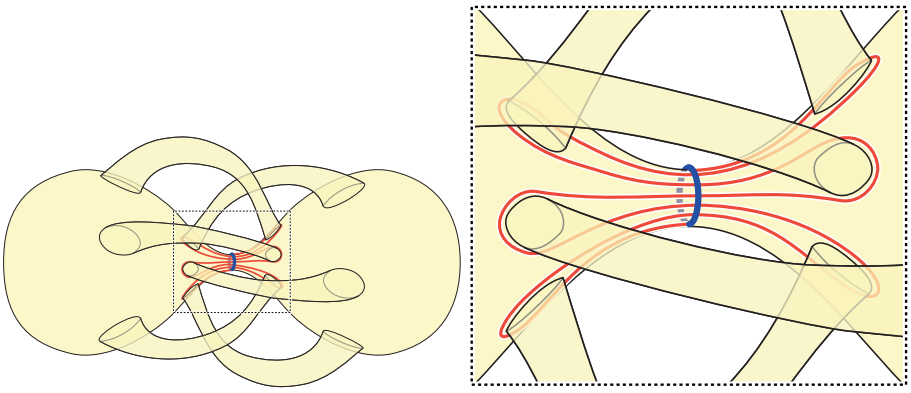

Figure 4.8. Ph'nglui mglw'nafh Cthulhu R'lyeh wgah'nagl fhtagn! A surface whose shortest splitting cycle cuts a shortest path g times, and a closeup of the undulating shortest splitting cycle.

The base surface $\mathcal{M}_0$ is a sphere whose geometry approximates an hourglass. Let γ be the central cycle that partitions the two lobes of the hourglass. To construct $\mathcal{M}_g$, we attach g handles to this hourglass, each joining a small circle c_i on the neck of the hourglass, just to one side of γ, to a large circle C_i far away on the opposite lobe. The small circles c_i are arranged symmetrically around the neck of the hourglass, alternating between the two sides of γ; the large circles C_i are also partitioned evenly between the two lobes of the hourglass.

Let μ be a cycle that undulates around the small circles in order, crossing γ a total of g times, as shown in Figure 4.8. We easily verify that μ splits $\mathcal{M}_g$ into two surfaces of genus $g/2$. Let $x_1, x_2, \ldots, x_g = x_0$ denote the g intersection points of μ and γ. For each i, let μ_i and γ_i respectively denote the subpaths of

[2] In general, this characterization requires shortest paths that terminate in the interior of edges, but if we refine edges appropriately, the shortest cycle will indeed be the concatenation of two shortest vertex-to-vertex paths.

μ and γ between x_{i-1} and x_i. Finally, we partition path γ_1 into three subpaths $\gamma_1^\flat$, $\gamma_1^\natural$, and $\gamma_1^\sharp$ at arbitrary points y and z. Let F denote the union of these $2g+2$ paths.

To obtain a combinatorial structure on $\mathcal{M}_g$, we embed a weighted graph G that contains F as a subgraph, such that every face of the embedding is a topological disk. We assign length 1 to edge γ_i for each $i \neq 1$, length 1 to edges $\gamma_1^\flat$ and $\gamma_1^\sharp$, length $g+1$ to edge $\gamma_1^\natural$, length $4g^2$ to each edge μ_i, and length at least $10g^5$ to every other edge in G. The cycle μ does not contain a single (even approximate!) vertex-to-vertex shortest path. Even if we allow shortest paths between points in the interior of edges, each such path contains at most one vertex x_i. The cycle γ can clearly be partitioned into two shortest paths of length $g+1$ at points y and z, and μ crosses one of these paths g times. Thus, to complete the proof, it suffices to show that μ is in fact the shortest splitting cycle in $\mathcal{M}_g$.

Let α be a shortest splitting cycle; the assigned edge weights guarantee that α is a cycle in F. By Lemma 4.3.1, α traverses each path γ_i and μ_i at most once in each direction. For any i, there is a path in $\mathcal{M}_g \setminus F$ from one side of γ_i to the other, so α must traverse each edge γ_i either twice (in opposite directions) or not at all.

Consider any simple cycle β in a tubular neighborhood $\mathcal{N}$ of F that traverses every edge in F either once in each direction or not at all. This cycle must be null-homologous, and therefore separating in $\mathcal{M}$ and $\mathcal{N}$. Furthermore, the boundary of $\mathcal{N}$ belongs entirely to some component of $\mathcal{M} \setminus \beta$. It follows that one component of $\mathcal{N} \setminus \beta$ is only bounded by β. This component must be a disk since $\mathcal{N}$ has genus zero. We conclude that β is contractible.

The splitting cycle α is not contractible, so it must traverse some edge μ_i exactly once. But α must traverse the edges adjacent to any vertex x_i an even number of times, which implies that α traverses *every* edge μ_i exactly once. Thus, every splitting cycle in F is at least as long as μ. We conclude that μ is indeed the unique shortest splitting cycle. □

4.4 Algorithm

In this section, we prove that computing the shortest splitting cycle is fixed-parameter tractable with respect to the genus and the number of boundary components of the surface.

Theorem 4.4.1. *Let $\mathcal{M}$ be an orientable cross-metric surface; let g be its genus, b be its number of boundary components, and n be its complexity. We can compute a shortest splitting cycle in $\mathcal{M}$ in $(g+b)^{O(g+b)} n \log n$ time.*

The algorithm proceeds in several stages, described in detail in the following subsections. First, we compute a set of $O(g+b)$ loops and arcs that cut the surface $\mathcal{M}$ into a disk, using (a variant of) the greedy algorithm of Erickson

and Whittlesey [59]; each loop and arc is the concatenation of two shortest paths. Next, we enumerate all possible sequences of crossings of this system of loops and arcs by a simple cycle that crosses each loop and arc $O(g + b)$ times. Proposition 4.3.2 implies that the shortest splitting cycle must have one of these crossing sequences. We discard any crossing sequence that does not correspond to a splitting cycle. For each valid crossing sequence, we compute a shortest cycle with that crossing sequence using the recent algorithm of Kutz [86]. The shortest of these cycles, γ, corresponds to the crossing sequence of a shortest splitting cycle; finally, we post-process γ to remove any self-intersections, without changing its length or its free homotopy type.

4.4.1 Greedy System of Loops or Arcs

We first compute a set of interior-disjoint shortest paths that split $\mathcal{M}$ into a disk.

If $\mathcal{M}$ has no boundary, let v be any point of $\mathcal{M}$ in the interior of a face of $G^*(\mathcal{M})$. Let $\alpha_1, \alpha_2, \ldots, \alpha_{2g}$ be the shortest system of loops of $\mathcal{M}$ with basepoint v; this system of loops can be computed in $O(n \log n + gn)$ time using a greedy algorithm of Erickson and Whittlesey [59].

A key property of this greedy system of loops is that each loop α_i is composed of two shortest paths in the primal graph $G(\mathcal{M})$. However, in general these two paths meet at a point m_i in the interior of some edge e_i. To simplify our algorithm, we split e_i into two edges at m_i—or equivalently, in the dual graph, we replace the dual edge e_i^* with two parallel edges—partitioning the length appropriately, so that α_i consists of two vertex-to-vertex shortest paths β_i and β_i' in $G(\mathcal{M})$. Another property that will be used later is that each α_i is a shortest loop in its homotopy class.

If $\mathcal{M}$ has at least one boundary component, then an easy variant on the aforementioned algorithm by Erickson and Whittlesey [59] allows us to compute a greedy *system of arcs*, which bears properties similar to the greedy system of loops: it is made of $O(g+b)$ arcs α_i; it can be computed in $O(n \log n + (g+b)n)$ time; each arc α_i is composed of two shortest paths β_i and β_i'; and each arc is as short as possible in its homotopy class. To compute the greedy system of loops with basepoint v, the idea is to maintain a set L, initially empty, of simple, pairwise disjoint loops. The algorithm iteratively adds to L the shortest loop based at v such that $\mathcal{M} \setminus L$ is connected, relying on a shortest paths tree rooted at the basepoint. To build the greedy system of arcs, the idea is similar: we iteratively add to the set of arcs L the shortest arc such that $\mathcal{M} \setminus L$ is connected. To implement this efficiently, it suffices to compute simultaneously shortest paths trees rooted at every boundary component.

4.4.2 Simple Crossing Sequences

The *crossing sequence* of a cycle γ records the intersections of γ with the greedy loops or arcs α_i, in cyclic order along γ. Any two cycles with the same crossing sequence are homotopic, although two homotopic cycles can have different crossing sequences. A crossing sequence is *simple* if it can be generated by a simple cycle; non-simple cycles can have simple crossing sequences.

Proposition 4.3.2 implies that some shortest splitting cycle γ crosses each path β_i or β_i' $O(g+b)$ times, and thus crosses each loop or arc α_i $O(g+b)$ times. Our algorithm enumerates a superset of all simple crossing sequences that cross each loop or arc α_i $O(g+b)$ times. There are $(g+b)^{\Theta((g+b)^2)}$ crossing sequences with $O(g+b)$ occurrences of each loop or arc, but the vast majority of these are not simple. Thus, to achieve our desired time bound, we cannot naively enumerate crossing sequences that satisfy Proposition 4.3.2 and then check each sequence for simplicity. Our enumeration algorithm enforces simplicity from the beginning.

We begin by cutting $\mathcal{M}$ along the loops or arcs α_i to obtain a polygonal schema $\mathcal{D}$; this is a cross-metric disk with complexity $O((g+b)n)$. This cutting operation also cuts the unknown splitting cycle γ into *segments* that cut across $\mathcal{D}$. Because γ is simple, no two of these segments cross. If $b = 0$, because the basepoint v does not lie on $G^*(\mathcal{M})$, we can slightly perturb γ without changing its length (in the crossing metric) or its homotopy class; thus, we assume that γ does not pass through the basepoint v. If $b \geq 1$, because no endpoint of a segment can be on an edge of the polygonal schema corresponding to a piece of boundary of $\mathcal{M}$, we contract these $4g+2b-2$ edges, working with a *contracted polygonal schema* with $4g+2b-2$ edges instead of $8g+4b-4$.

The segments of γ can be grouped into subsets according to which pair of greedy loops they meet on the boundary of $\mathcal{D}$ (Figure 4.9). We abstract and dualize the (contracted) polygonal schema by replacing each edge of the (contracted) polygonal schema with a vertex and connecting vertices that correspond to consecutive edges. Now each subset of segments corresponds to a diagonal between two vertices of the dual $4g$-gon (if $b = 0$) or $(4g+2b-2)$-gon (if $b \neq 0$). Since no two segments cross, these diagonals cannot cross. In particular, all the diagonals belong to some triangulation of the dual polygon.

Thus the candidate crossing sequences of a shortest splitting cycle are described by *weighted triangulations*, which consist of a triangulation of the dual polygon, each of whose edges is weighted with an integer between 0 and $O(g+b)$. The label of an edge in the triangulation represents the number of times that the cycle runs along that edge. There are C_{k-2} possible triangulations of a k-gon, where $C_k = O(4^k)$ is the kth Catalan number, which we can enumerate in $O(k)$ time each. (This is identical to the enumeration of binary trees, see Section 2.3.4.4 in [83].) Here $k = O(g+b)$. There are $(g+b)^{O(g+b)}$ ways to label each triangulation, which we can enumerate in constant amortized time

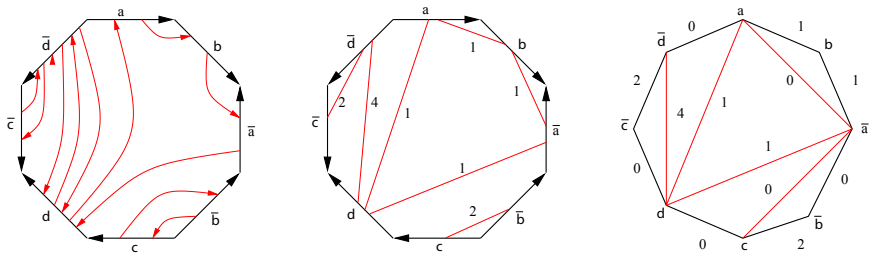

Figure 4.9. Left: A splitting cycle on a double-torus ($g = 2$, $b = 0$). Middle: The corresponding subsets of segments; each label indicates the number of segments contained in a subset. Right: The corresponding weighted triangulation.

per labeling.

We thus obtain a total of $(g+b)^{O(g+b)}$ weighted triangulations. Most of these do not correspond to a splitting cycle, or indeed to *any* cycle. We now explain how to discard these possibilities.

4.4.3 Testing Weighted Triangulations

Given a candidate weighted triangulation T, we want to test whether T corresponds to a splitting cycle. We must check that (1) T corresponds to a set of cycles, (2) this set contains exactly one cycle, and (3) this cycle is separating but noncontractible. (We already know that the cycle is simple.) We describe how to perform these tests in $O((g+b)^2)$ time. If any of these properties is not satisfied, we simply discard T.

To check that T corresponds to a set of cycles, it suffices to check that the two edges of $\mathcal{D}$ that correspond to the same α_i are crossed the same number of times. (If $\mathcal{M}$ has a boundary, no edge of $\mathcal{D}$ corresponding to a part of the boundary of $\mathcal{M}$ can be crossed, since we worked in the contracted polygonal schema.)

For the remaining tests, we build a combinatorial surface $\mathcal{M}'$ homeomorphic to $\mathcal{M}$, whose graph G' is the arrangement of the greedy system of loops and the candidate cycle(s) defined by T. We cut the abstract (non-contracted) polygonal schema along the subsets of edges given by the triangulation and then identify corresponding subpaths on the boundary of the polygonal schema. If we have multiple edges running along the same edge of the triangulation, these define thin rectangular strips. The complexity of the resulting surface is $O(g+b)^2$. The $O(n)$ internal complexity of the original surface $\mathcal{M}$ is ignored.

Once we have constructed $\mathcal{M}'$, we can check whether T defines a single cycle γ by a simple depth-first search.

To test that γ is separating but noncontractible, we use a simplification of an algorithm of Erickson and Har-Peled [58]. To test separation, we perform a depth- or breadth-first search on the faces of $\mathcal{M}'$, starting from any initial face, but forbidding crossings of any segment of γ. The cycle γ is separating if and only if the search halts before visiting every face of the surface. (Alternately, γ

is separating if and only if γ crosses each α_i the same number of times in both directions.) We also compute the Euler characteristic of the reachable portion of the surface during the search by counting vertices, edges, and faces. The cycle is noncontractible if and only if this Euler characteristic is neither 1 (a disk) nor $1 - 2g - b$ (the complement of a disk).

4.4.4 From Crossing Sequence to Cycle

For each valid weighted triangulation, we compute the shortest cycle with the corresponding crossing sequence in time $O((g+b)^3 n \log n)$ using the recent algorithm of Kutz [86]. For the sake of completeness, we sketch the algorithm here.

First we glue together a cycle of $O((g+b)^2)$ distinct copies of the polygonal schema $\mathcal{D}$, one per crossing in the sequence. Each successive pair of copies is glued along the edge specified by the corresponding entry in the crossing sequence. Because we consider only crossing sequences without *spurs*—the cycle does not cross a loop α_i and then immediately recross the same loop α_i in the opposite direction—the resulting combinatorial surface is an annulus, which we denote $\mathcal{D}^\circ$. (In Kutz's terms, we are considering only *curl-free* splitting cycles.) The polygonal schema $\mathcal{D}$ has complexity $O((g+b)n)$, so the complexity of $\mathcal{D}^\circ$ is $O((g+b)^3 n)$.

We then compute the shortest cycle γ° in $\mathcal{D}^\circ$ that is freely homotopic to the boundaries of $\mathcal{D}^\circ$, using an algorithm of Frederickson [61]; see also [33, Lemma 3.3(d)]. Given a combinatorial annulus of complexity N, Frederickson's algorithm finds a shortest generating cycle in $O(N \log N)$ time. Thus, we compute γ° in $O((g+b)^3 n \log((g+b)^3 n)) = O((g+b)^3 n \log n)$ time.

Finally, projecting γ° back to the original surface $\mathcal{M}$ gives us a shortest cycle γ with the given crossing sequence.

Since there are $(g+b)^{O(g+b)}$ valid crossing sequences, the total time spent in this phase is $(g+b)^{O(g+b)} O((g+b)^3 n \log n) = (g+b)^{O(g+b)} n \log n$.

4.4.5 Removing Self-Intersections

Let γ be the shortest cycle computed in the previous phase, over all possible valid weighted triangulations. This cycle is null-homologous, noncontractible, as short as possible in its homotopy class, and freely homotopic to a simple cycle, but *not* necessarily simple. Although the cycle γ° is simple, projecting it back to the original surface may introduce self-intersections. We will prove that some simple cycle γ' is homotopic to γ and has the same length as γ (this also follows from [34]); hence, γ has the correct homotopy class of a shortest splitting cycle. We then need to compute such a γ'.

We consider the polygonal schema defined by the greedy loops or arcs. A *segment* is a path that intersects the polygonal schema precisely at its endpoints. A set of segments S *respects* a weighted triangulation T if, for each pair of edges

of the polygonal schema, the number of segments of S between them equals the weight on the corresponding edge of T (or zero if there is no such edge). We need the following lemma.

Lemma 4.4.2. *Let T be a weighted triangulation and let S be a set of segments that respects T. There is a set of simple, pairwise disjoint segments of total length no larger than the total length of S that respects T and intersects the greedy loops or arcs at the same points as S.*

Proof: We prove the result by induction on the number of crossings in S. If S has no crossings, there is nothing to show. Otherwise, we will transform S into another set of segments that has one or two fewer crossings, still respects T, and intersects the greedy loops or arcs at the same points.

Consider a crossing in S in the polygonal schema. If this corresponds to a self-crossing of a single segment, we may remove the monogon based at this crossing point and conclude by the induction hypothesis.

Otherwise, the crossing corresponds to two segments s and s'. If these segments cross at least twice, they form a bigon (two simple interior-disjoint paths with the same endpoints) inside the polygonal schema, which can be flipped to obtain a set of segments respecting T that is no longer and has two crossings fewer.

Otherwise, s and s' cross exactly once, at some crossing point p; they must thus correspond to (possibly identical) edges e and e' of the triangulation. In a triangulation, no two edges cross; if e and e' do not share any endpoint, then s and s' would cross an even number of times, which is not the case; so e and e' must share at least one endpoint, which is also an edge E of the polygonal schema. Hence swapping the parts of s and s' that are between E and p also results in a set of segments respecting T; this operation decreases the number of crossings by one and does not increase the length. □

Recall that γ is the cycle computed in the previous section. Let T be the weighted triangulation corresponding to γ; it was also computed above.

We apply the following algorithm. Consider the intersections of the cycle γ with the greedy loops or arcs. There is exactly one way to connect these points with simple, disjoint segments respecting T. We iteratively connect these pairs of points by shortest segments, forbidding, at each step, any crossing with the previously computed segments. By a simple exchange argument, the segments created are shortest paths in the polygonal schema; so this set of segments is a shortest set of segments, among all sets of simple, pairwise disjoint segments respecting T and intersecting the polygonal schema at the same points as γ. By Lemma 4.4.2, this set of segments is no longer than the set of segments of γ. Since it is a set of simple, pairwise disjoint segments respecting the valid triangulation T, it forms a *single* cycle γ', which has the same crossing sequence as γ by construction of γ. So γ' is the desired splitting cycle.

There are $O((g+b)^2)$ shortest paths to compute, in a planar graph of complexity $O((g+b)n)$. The cost of this uncrossing step is thus $O((g+b)^3 n)$. Using a result by Takahashi, Suzuki, and Nishizeki [113, Section 3], we can improve the running time to $O((g+b)n \log (g+b))$.

This concludes the proof of Theorem 4.4.1.

4.5 Splitting Surfaces into Two Surfaces of Prescribed Topology

Let $\mathcal{M}$ be a surface with genus g and with b boundary components. A splitting cycle splits $\mathcal{M}$ into two surfaces $\mathcal{M}_1$ and $\mathcal{M}_2$, with respective genus g_1 and g_2 and with respective number of boundary components $b_1 \geq 1$ and $b_2 \geq 1$, where $g_1 + g_2 = g$ and $b_1 + b_2 = b + 2$. The techniques developed in the previous section allow to compute the shortest splitting cycle that splits the surface into two surfaces of prescribed genus and number of boundaries.

More precisely, let $S = \{(g_1, b_1), \ldots, (g_k, b_k)\}$ be a set of ordered pairs of integers. We say that a splitting cycle on $\mathcal{M}$ is *allowed* by S if it splits $\mathcal{M}$ into two surfaces, one of which has genus g' and b' boundary components, for some $(g', b') \in S$. In particular, if $S = (\{0, \ldots, g\} \times \{1, \ldots, b+1\}) \setminus \{(0,1), (0,2), (g,b), (g,b+1)\}$, the splitting cycles allowed by S are precisely the essential splitting cycles (the splitting cycles that do not bound an annulus).

Theorem 4.5.1. *Let $\mathcal{M}$ be an orientable cross-metric surface with genus g and b boundary components; let S be a set of ordered pairs. We can compute a shortest splitting cycle allowed by S in $(g+b)^{O(g+b)} n \log n$ time.*

Proof: If S contains $(0,0)$ or $(g, b+1)$, then a contractible cycle of length zero is a shortest splitting cycle allowed by S. Otherwise, the algorithm described in the previous sections, with very minor modifications, also works. Lemma 4.3.1 and Proposition 4.3.2 extend verbatim to the case where we are looking for a shortest splitting cycle allowed by S, because their proofs are based on exchange arguments that change the splitting cycle without modifying the topology of the surface on either side of the cycle. The algorithm has to be modified in Section 4.4.3 because we now have to check that a given weighted triangulation corresponds to a splitting cycle that is allowed by S; this is straightforward and does not increase the running-time of the algorithm. □

4.6 Decomposition into Punctured Tori

A *decomposition into punctured tori* of an orientable surface $\mathcal{M}$ with genus g and no boundary is a set of $g-1$ pairwise disjoint, pairwise non-homotopic, splitting cycles. Equivalently, it is a set of simple, pairwise disjoint cycles that split $\mathcal{M}$ into g punctured tori.

There is a naïve greedy algorithm to compute a decomposition into punctured tori: compute the shortest splitting cycle of $\mathcal{M}$, cut along it, fill the boundaries obtained with a disk, and recurse in each of the two resulting surfaces.

Proposition 4.6.1. *The greedy algorithm does not necessarily provide the shortest decomposition into punctured tori.*

Proof: We use the graph G, embedded in a triple-torus, shown on Figure 4.10. The graph is extended, with edges of very large weight, to obtain a combinatorial surface $\mathcal{M}$. Any decomposition into punctured tori of $\mathcal{M}$ consists of precisely two cycles.

There is a decomposition into punctured tori of $\mathcal{M}$ of length 28, see Figure 4.11: one cycle is $\bar{b}ab\bar{c}\bar{a}c$, the other one is $fh\bar{f}g\bar{h}\bar{g}$. It is actually easy to prove that it is the shortest decomposition into punctured tori, but we won't need that. We will prove that the greedy algorithm gives the decomposition into punctured tori depicted in Figure 4.12, of length 30.

G can be decomposed into five edge-disjoint cycles that are homologically independent in $\mathcal{M}$ and form a cycle basis of G. It follows that the shortest splitting cycle uses two of these cycles at least twice. From this remark and from the assignment of the weights of G, we deduce that there are exactly two shortest splitting cycles, $bd\bar{e}\bar{b}ce\bar{d}\bar{c}$ and its symmetric, $\bar{f}\bar{d}ef\bar{g}\bar{e}dg$, on $\mathcal{M}$; they both have length 8. Without loss of generality, assume that the greedy algorithm chooses the first one, γ, as first cycle of the decomposition into punctured tori.

Now, cutting $\mathcal{M}$ along γ and filling its boundaries with disks yields two connected surfaces, a torus and a double-torus. The double-torus will be split with another splitting cycle; it is a combinatorial surface $\mathcal{M}'$, as shown on Figure 4.13, left.

Consider the surface $\mathcal{M}''$ on Figure 4.13, right. It is the same surface as $\mathcal{M}'$, except that the pairs of edges $b'd'$ and $b''e'$, and $c'd''$ and $c''e''$, have been merged and the leftmost endpoints of these edges have been identified. Any splitting cycle on $\mathcal{M}'$ gives a splitting cycle on $\mathcal{M}''$, of the same length. The graph on $\mathcal{M}''$ is made of three homologically independent cycles; arguing as above, the shortest splitting cycle on $\mathcal{M}''$ uses two cycles at least twice, and thus must have length at least 22. Hence the shortest splitting cycle on $\mathcal{M}'$ has length at least 22; but, as shown on Figure 4.12, there is such a cycle. Hence the greedy algorithm provides a decomposition into punctured tori of total length 30. □

4.7 Conclusions

The results of this chapter suggest several open problems. Most notably, can we approximate the shortest splitting cycle, or is that also NP-hard? The following high-level approach seems promising. Compute shortest simple cycles in each

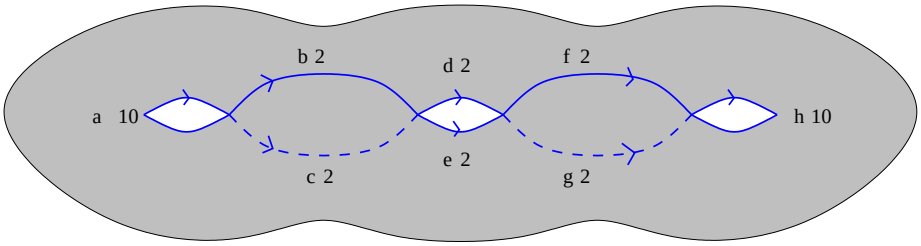

Figure 4.10. The graph G embedded on a triple-torus $\mathcal{M}$ in the proof of Proposition 4.6.1. The weights of the edges of G are indicated in parentheses.

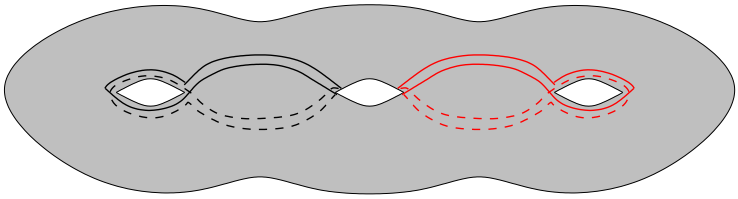

Figure 4.11. The shortest decomposition into punctured tori, of length 28.

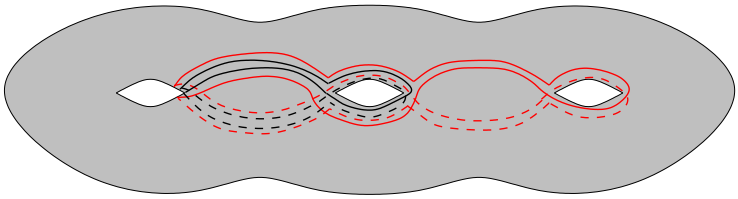

Figure 4.12. The greedy decomposition into punctured tori, of length 30. The first cycle has length 8, the second one has length 22.

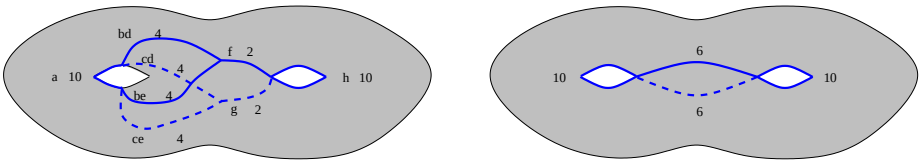

Figure 4.13. Left: the double-torus $\mathcal{M}'$ obtained after cutting $\mathcal{M}$ along the shortest splitting cycle and filling the boundary with a disk. Each edge b, c, d, and e on $\mathcal{M}$ corresponds to two edges on $\mathcal{M}'$. The weights of the edges are indicated. Right: the surface $\mathcal{M}''$.

non-trivial homotopy class in order of increasing length, stopping either when we find a separating cycle, or when we find two cycles α and β that intersect an odd number of times. If we find a separating cycle, it is of course the shortest splitting cycle. If we find two cycles with odd intersection number, an exchange argument implies that the intersection number is 1. For some orientation of α and β, the cycle $\alpha \cdot \beta \cdot \bar{\alpha} \cdot \bar{\beta}$ is a splitting cycle whose length is at most four times the length of the shortest splitting cycle. Can this algorithm be implemented efficiently? How quickly can we enumerate the k shortest homotopy classes of (simple) cycles? Techniques of Eppstein [56] for enumerating k shortest paths may be useful here.

If we iteratively cut along splitting cycles on the surface where each boundary is filled with a disk, we obtain a decomposition of the surface into punctured tori. However, we proved that repeatedly cutting the surface along its shortest splitting cycle does not necessarily yield the shortest such decomposition. Does this algorithm provide a constant-factor approximation for this problem? Is computing the shortest torus decomposition NP-hard? Similarly, a *pants decomposition* is a set of disjoint simple cycles decomposing a surface into pairs of pants, or spheres with three boundary components [34]. Is it NP-hard to compute the shortest pants decomposition? Are either of these problems fixed-parameter tractable?

Finally, Erickson and Har-Peled [58] prove that finding the minimum cut graph is NP-hard, by a reduction from the fixed-parameter tractable rectilinear Steiner tree problem. Is computing the shortest cut graph fixed-parameter tractable? The most serious bottleneck here seems to be computing the shortest cut graph in a given homotopy class, where the homotopy is allowed to move the vertices.

Chapter 5

Homotopic Fréchet Distance

Given two curves on a surface, it is natural to ask how similar they are. In previous sections, we have discussed questions relating to homotopy, but this is not always the best classification for similarity. Two curves on a surface may be homotopic but be far apart from each other or be separated by "mountains", or areas of high curvature.

One common measure of similarly between curves is the Hausdorff distance. For two curves A and B, the Hausdorff distance between them is equal to $\max_{a in A} \min_{b \in B} d(a, b)$. While the Hausdorff metric does measure closeness in space, it does not take into account the flow of the curves, which in many applications, such as morphing in computer graphics, is an important property of the curves.

The *Fréchet distance*, sometimes called the *dog-leash distance*, is defined as the minimum length of a leash required to connect a dog and its owner as they walk without backtracking along their respective curves from one endpoint to the other. The Fréchet metric takes into account the flow of the two curves because the pairs of points whose distance contributes to the Fréchet metric sweep continuously along their respective curves. It is therefore possible for two curves to have small Hausdorff distance but large Fréchet distance. Fréchet distance is used as a more accurate measure of similarity in applications such as hand writing recognition [110].

For piecewise linear curves in Euclidean space, an algorithm to compute Fréchet distance is known [3, 2]. Other citations.

The definition of Fréchet distance accommodates any underlying metric, and thus can be generalized to other types of metric spaces. For instance, if the two curves lie on a terrain (or any surface) [89], geodesic distance is a natural choice, since the leash must lie on the surface. Cook et. al. develop algorithms to compute *geodesic Fréchet distance*, where the leash is constrained to the interior of a simple polygon but remains a geodesic curve at all times [36, 53]. In both settings, the leash is allowed to move discontinuously from one side of an obstacle or a mountain to another.

In this chapter, we introduce a continuity requirement on the motion of the leash. We require that the leash move continuously as the dog and his owner walk along the curves; in particular, the leash cannot jump over obstacles and

can sweep over a mountain only if it is long enough. We define the *homotopic Fréchet distance* between two curves as the Fréchet distance with this additional continuity requirement. This continuity requirement is satisfied automatically for curves inside a simple polygon, but not in more general environments like convex polyhedra.

The motion of the leash defines a correspondence between the two curves that can be used to morph between the two curves—two points joined by a leash morph into each other. The homotopic Fréchet distance describes a morphing which minimizes the maximum amount of deformation needed to transform one curve into the other. See [54] for similar results on morphing.

Efficiently computing the homotopic Fréchet distance in general metric spaces is a new open problem. We present a polynomial-time algorithm for a special case of this problem, which is to compute the homotopic Fréchet distance between two polygonal curves in the plane minus a set of point or polygonal obstacles. Note that the curves themselves could also be obstacles in this setting.

5.1 Definitions

Let S be a fixed Hausdorff metric space. A *curve* in S is a continuous function from the unit interval $[0,1]$ to S. We say a curve is a *geodesic* if it is locally as short as possible. We will sometimes abuse notation by using the same symbol to denote a curve $A\colon [0,1] \to S$ and its image in S. A *reparameterization of* $[0,1]$ is a continuous, non-decreasing, surjection $\alpha\colon [0,1] \to [0,1]$. A reparameterization of a curve $A\colon [0,1] \to S$ is any curve $A \circ \alpha$, where α is a reparameterization of $[0,1]$. The *length* of any curve A, denoted $\text{len}(A)$, is defined by the metric of S; in particular, two reparameterizations of the same curve have the same length.

A *leash* between two curves A and B is another curve $\lambda\colon [0,1] \to S$ such that $\lambda(0) = A(s)$ and $\lambda(1) = B(t)$ for some parameters s and t. A *homotopy* between curves A and B is a continuous map $h\colon [0,1] \times [0,1] \to S$ such that $h(\cdot, 0) = A$ and $h(\cdot, 1) = B$. For any $t \in [0,1]$, the one-parameter function $h(t, \cdot)$ is a leash from A to B. A *leash map* between curves A and B is a homotopy between some reparameterization of A and some reparameterization of B. Intuitively, a leash map describes the continuous motion of a leash between a dog walking along A and its owner walking along B. The *length* of a leash map ℓ, denoted $\text{len}(\ell)$, is the maximum length of any leash $\ell(t, \cdot)$. Finally, the *homotopic Fréchet distance* between two curves A and B, denoted $\overline{\mathcal{F}}(A, B)$, is the infimum, over all leash maps ℓ between A and B, of the length of ℓ:

$$\overline{\mathcal{F}}(A,B) := \inf_{\text{leash map } \ell:[0,1]^2 \to S} \left(\max_{0 \le t \le 1} \text{len}(\ell(t,\cdot)) \right).$$

In contrast, the classical ("leashless") Fréchet distance is defined directly in

terms of reparameterizations and distances:

$$\mathcal{F}(A,B) := \inf_{\alpha,\beta:[0,1]\to[0,1]} \left(\max_{0\le t\le 1} d(A(\alpha(t)), B(\beta(t))) \right).$$

In spaces where shortest paths vary continuously as their endpoints move, such as Euclidean space or the interior of a simple polygon, the two definitions are equivalent. In general, however, the homotopic Fréchet distance between two curves can be larger (but never smaller) than the classical Fréchet distance.

A *homotopy relative to A and B*, or simply *relative homotopy*, is a continuous function $h\colon [0,1]\times[0,1] \to S$, such that $h(\cdot,0)$ and $h(\cdot,1)$ are respectively of the form $A(u(\cdot))$ and $B(v(\cdot))$, where u and v are re-parameterizations of $[0,1]$. Two leashes λ and λ' are *relatively homotopic*, denoted $\lambda \simeq \lambda'$, if there is a relative homotopy h such that $h(0,\cdot) = \lambda$ and $h(1,\cdot) = \lambda'$. It is easy to prove that $\simeq$ is an equivalence relation over the set of leashes from A to B. Any leash map is (the transpose of) a relative homotopy; thus, all leashes $\ell(t,\cdot)$ determined by a leash map ℓ lie in the same relative homotopy class.

5.2 Preliminaries

In this paper, we develop a polynomial-time algorithm to compute the homotopic Fréchet distance between two polygonal paths in $\mathcal{E} = \mathbb{E}^2 \setminus P$, for some set P of closed polygons, where we consider geodesic distance with an underlying L_2 metric. The polygons P act as obstacles; in any leash map in $\mathcal{E}$, the moving leash can neither touch nor jump over any obstacle. Note that the polygons themselves can be a part of P.

Specifically, the input to our problem consists of two polygonal curves A and B and a set P of polygonal objects in the Euclidean plane. Curves A and B may (self-)intersect, but neither curve intersects any obstacle in P. To simplify our exposition, we assume that no three vertices of the input (vertices of polygons in P or vertices of A and B) are collinear; this assumption can be enforced algorithmically using standard perturbation techniques [107]. A and B may be elements of P, or not.

Let $a_0, a_1, \ldots, a_m$ denote the sequence of vertices of A; these points define a standard parameterization $A\colon [0,m] \to \mathcal{E}$ whose restriction to any integer range $[i-1, i]$ is an affine map onto the corresponding edge $a_{i-1}a_i$. Similarly, the vertices $b_0, b_1, \ldots, b_n$ of B define a piecewise-affine parameterization $B\colon [0,n] \to \mathcal{E}$. Let $P_1, P_2, \ldots, P_o$ denote the obstacle polygons in P, and let k denote the total number of vertices in all obstacle polygons. In the special case where every obstacle is a single point, we obviously have $k = o = |P|$. Finally, let $N = n + m + k + 2$ denote the total complexity of the input.

Figure 5.1 illustrates optimum leash maps for a few sample inputs where P is a discrete set of points.

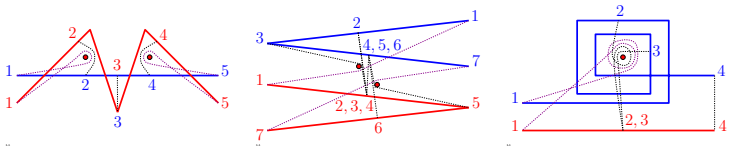

Figure 5.1. Optimum leash maps for three inputs. Dashed curves between matching numbers represent intermediate leashes.

5.2.1 Geodesic Leash Maps

To simplify our presentation, we will allow 'paths' in $\mathcal{E}$ to touch obstacles in P. Specifically, we consider *geodesics*: piecewise-linear curves in the closure of $\mathcal{E}$, whose interior vertices are vertices in P. Although geodesics may run along obstacle boundaries, they do not intersect the interior of any obstacle.

In the special case where the obstacles are points (so the closure of $\mathcal{E}$ is the entire plane), we need some additional information to ensure that each geodesic lies in a unique homotopy class. Specifically, we associate a *turning angle* with each obstacle point that a geodesic touches. Let C_ε be a circle centered at obstacle point p and radius ε, small enough to exclude every other obstacle in P. A turning angle of θ at an obstacle point p indicates that replacing the portion of γ inside C_ε with a counterclockwise arc of length $\theta\varepsilon$ around C_ε yields a new path homotopic to γ. For example, a path with turning angle zero makes a U-turn at p without enclosing p; a path that goes straight through p with p on its left (resp. right) has turning angle π (resp. $-\pi$); a turning angle of 10π means the path makes a U-turn after winding around the point five times counterclockwise. A geodesic could meet the same obstacle point more than once; we associate a different turning angle with each incidence.

It can be shown that for any two points x and y in $\mathcal{E}$, the shortest path from x to y in any relative homotopy class is a unique geodesic, in which every turning angle at a point obstacle is either at least π or at most $-\pi$. Conversely, every geodesic in which every turning angle is either at least π or at most $-\pi$ is a shortest path in some homotopy class.

A *geodesic leash map* is a leash map $\ell\colon [0,1] \times [0,1] \to \mathbb{E}^2$ in which every leash $\ell(t,\cdot)$ is a geodesic, and all these geodesics are in the same relative homotopy class. We next prove that geodesic leash maps are optimal within their homotopy class.

Lemma 5.2.1. *Suppose ℓ is a leash map between two curves A and B. There is a geodesic leash map ℓ' between A and B such that, for all $t \in [0,1]$, the leash $\ell'(t,\cdot)$ is the shortest path homotopic to $\ell(t,\cdot)$ with the same endpoints. Additionally, the length of ℓ' is at most the length of ℓ.*

Proof: We lift ℓ to the universal cover $\hat{\mathcal{E}}$ of $\mathcal{E}$, obtaining a leash map $\hat{\ell}$ between the lifts $\hat{A}$ and $\hat{B}$ of A and B respectively. For each $t \in [0,1]$, let $\hat{\ell}'(\cdot,t)$ be the shortest path between the endpoints of $\hat{\ell}(t,\cdot)$ in the same homotopy class. The

universal cover $\hat{\mathcal{E}}$ is a simply-connected space with a locally Euclidean metric, so shortest paths in $\hat{\mathcal{E}}$ vary continuously as the endpoints move continuously. It follows that $\hat{\ell}'$ is a continuous function in both arguments, and therefore a (geodesic) leash map in $\hat{\mathcal{E}}$. The projection ℓ' of $\hat{\ell}'$ back to $\mathcal{E}$ is a (geodesic) leash map between A and B. For all t, the leash $\ell'(t,\cdot)$ is the shortest path homotopic to $\ell(t,\cdot)$, so $\max_t \operatorname{len}(\ell'(t,\cdot)) \leq \max_t \operatorname{len}(\ell(t,\cdot))$. □

Lemma 5.2.1 implies that the homotopic Fréchet distance between A and B is the length of a *geodesic* leash map in some homotopy class determined by some reparameterizations of A and B. Thus, the homotopic Fréchet distance can be redefined as the minimum, over all homotopy classes h, of the *classical* Fréchet distance, where distances are defined by shortest paths in relative homotopy class h:

$$\mathcal{F}_h(A,B) := \min_{\alpha,\beta:[0,1]\to[0,1]} \left(\max_{t\in[0,1]} D_h(A(\alpha(t)), B(\beta(t))) \right)$$

$$\overline{\mathcal{F}}(A,B) := \min_{\text{homotopy class } h} \mathcal{F}_h(A,B)$$

(Here, $D_h(u,v)$ denotes the length of the shortest path from u to v in relative homotopy class h.)

We call a relative homotopy class h *optimal* if $\overline{\mathcal{F}}(A,B) = \mathcal{F}_h(A,B)$.

For the rest of the paper, we restrict ourselves to geodesic leashes and geodesic leash maps. In Section 5.3, we provide a characterization of an optimal homotopy class, and we use this characterization to enumerate possible optimal homotopy classes in polynomial time. In Section 5.4, we describe a polynomial-time algorithm to compute the Fréchet distance within a particular homotopy class. Combining these two subroutines gives us a polynomial-time algorithm to compute homotopic Fréchet distance.

5.2.2 Homotopic Shortest Paths

Our algorithm relies on observations by Hershberger and Snoeyink [77] about shortest homotopic paths in the punctured plane; see also [66, 18, 55, 9, 10]. Suppose we already know a shortest path (a leash) λ between points $a \in A$ and $b \in B$, such as, for instance, a straight-line segment ab. To compute the geodesic leash between some other pair of points in the same homotopy class as λ, we follow the continuous evolution of the geodesic as the points a and b move along their respective curves. The sequence of obstacle vertices on the leash behaves like a *double-ended queue* or *deque*. A new vertex is pushed onto one end of the deque whenever the first or last segment of λ collides with an obstacle vertex. Conversely, a vertex is popped off one end of the deque when the first or last two segments of λ become collinear, and if their common vertex is a point obstacle, the turning angle at that point is either π or $-\pi$.

5.3 Optimal Homotopy Classes

5.3.1 Minimality

Let $\text{len}(\lambda)$ denote the length of any geodesic leash λ, and let $\text{turn}(\lambda)$ denote the sum of the absolute values of the turning angles at any point obstacles on λ. (Again, if λ meets the same point obstacle more than once, each incidence separately contributes to $\text{turn}(\lambda)$. If none of the obstacles are points, then $\text{turn}(\lambda) = 0$.) For any pair of leashes λ and λ', we write $\lambda \preceq \lambda'$ if and only if either (a) $\text{len}(\lambda) < \text{len}(\lambda')$, or (b) $\text{len}(\lambda) = \text{len}(\lambda')$ and $\text{turn}(\lambda) \leq \text{turn}(\lambda')$. We write $\lambda \prec \lambda'$ whenever $\lambda \preceq \lambda'$ but $\lambda' \not\preceq \lambda$.

We can extend this partial order to homotopy classes as follows. For any relative homotopy class h and any $s, t \in [0, 1]$, let $\sigma_h(s, t)$ denote the shortest path in h between points $A(s)$ and $B(t)$. For any two homotopy classes h and h', we write $h \preceq h'$ if and only if $\sigma_h(s, t) \preceq \sigma_{h'}(s, t)$ for all parameters s and t. We write $h \prec h'$ whenever $h \preceq h'$ but $h' \not\preceq h$.

Lemma 5.3.1. *For any relative homotopy classes h and h', if $h \preceq h'$, then $\mathcal{F}_h(A, B) \leq \mathcal{F}_{h'}(A, B)$.*

Proof: Let ℓ' be an optimum leash map in homotopy class h', so that $\text{len}(\ell') = \mathcal{F}_{h'}(A, B)$. For some reparameterizations α and β, we have $\ell'(t, \cdot) = \sigma_{h'}(\alpha(t), \beta(t))$ for all t. Let ℓ be the geodesic leash map in homotopy class h defined by the same reparameterizations: $\ell(t, \cdot) = \sigma_h(\alpha(t), \beta(t))$ for all t. The definition of $\preceq$ implies that $\text{len}(\ell(t, \cdot)) \leq \text{len}(\ell'(t, \cdot))$ for all t. It follows that $\mathcal{F}_h(A, B) \leq \text{len}(\ell) \leq \text{len}(\ell') = \mathcal{F}_{h'}(A, B)$. □

A relative homotopy class h is *minimal* if $h' \preceq h$ implies $h \preceq h'$.

Lemma 5.3.2. *For any relative homotopy class h, there is a minimal relative homotopy class h' such that $h' \preceq h$.*

Proof: Assume, for the sake of contradiction, that there is no minimal relative homotopy class h' such that $h' \preceq h$. Then we can construct an arbitrarily long descending chain of relative homotopy classes $h = h_0 \succ h_1 \succ h_2 \succ \cdots$. To simplify notation, let $\sigma_n = \sigma_{h_n}(0, 0)$.

Consider the ordered list of obstacle points on each path σ_n. There are finitely many such ordered lists, because $\text{len}(\sigma_n) \leq \text{len}(\sigma_0)$ for each n. Thus, up to taking a subsequence, we may assume that every path σ_n visits the same sequence of obstacle points. This assumption implies that all paths σ_n are geometrically equivalent and thus have equal length. Thus, by definition of $\preceq$, we have $\text{turn}(\sigma_n) < \text{turn}(\sigma_0)$ for all n. There are finitely many relative homotopy classes with a given ordered list of vertices and with bounded total absolute turning angle. (Specifically, since $\text{turn}(\sigma_n) - \text{turn}(\sigma_0)$ is always a multiple of 2π, there are at most $\lfloor \text{turn}(\sigma_0)/2\pi \rfloor + 1$ such classes.) □

The two previous lemmas immediately imply that there is a *minimal* optimal homotopy class.

In the next two subsections, we characterize minimal homotopy classes and describe how to enumerate them efficiently, first for point obstacles and then for polygonal obstacles.

5.3.2 Point Obstacles

Suppose P is a fixed finite set of points. A *proper line segment* is a line segment in $\mathcal{E}$ (and thus disjoint from P) joining a point in A to a point in B.

Proposition 5.3.3. *A relative homotopy class is minimal if and only if it contains a proper line segment.*

Proof: One direction of the proof is straightforward. Let h be the relative homotopy class of the proper line segment σ from $A(s)$ to $B(t)$. For any relative homotopy class $h' \neq h$, the shortest path $\sigma_{h'}(s,t)$ must be longer than σ, so $\sigma_{h'}(s,t) \not\preceq \sigma = \sigma_h(s,t)$, which implies that $h' \not\preceq h$. We conclude that h is minimal.

Now let h be an arbitrary minimal homotopy class. Let $\hat{A}$ and $\hat{B}$ be lifts of A and B in the universal cover $\hat{\mathcal{E}}$, such that for all s and t, the shortest path $\hat{\sigma}_h(s,t)$ between $\hat{A}(s)$ and $\hat{B}(t)$ is a lift of $\sigma_h(s,t)$. Let $\hat{P}$ denote the set of *all* lifts of points in P; these lifted obstacle points lie on the boundary of $\hat{\mathcal{E}}$.

We prove that h contains a proper line segment in two stages. First, we prove that no lifted obstacle point $\hat{p} \in \hat{P}$ lies on *every* path $\hat{\sigma}_h(s,t)$. Next, we construct a relative homotopy from the initial leash $\sigma_h(0,0)$ to a proper line segment.

Stage 1: No common corner. For the sake of deriving a contradiction, suppose there is a lifted obstacle point $\hat{p} \in \hat{P}$ such that for all s and t, the path $\hat{\sigma}_h(s,t)$ passes through $\hat{p}$. For all s and t, the path $\hat{\sigma}_h(s,t)$ is a shortest path, so its turning angle at $\hat{p}$ must lie outside the open interval $(-\pi, \pi)$. This turning angle is a continuous function of s and t, so we can assume without loss of generality that it is always greater than π. In other words, we assume that every path $\hat{\sigma}_h(s,t)$ winds counterclockwise around $\hat{p}$.

Now $\hat{p}$ is a lift of some obstacle $p \in P$, and $\hat{\sigma}_h(s,t)$ similarly projects to a geodesic $\sigma_h(s,t)$. For each s and t, let $\tau(s,t)$ denote the path with the same vertices and turning angles as $\sigma_h(s,t)$, except that the turning angle at p is reduced by 2π. All paths $\tau(s,t)$ belong to a single relative homotopy class, which we denote h'.

Fix parameters s and t, and consider the turning angles of $\sigma_h(s,t)$ and $\tau(s,t)$ at p. If the turning angle of $\sigma_h(s,t)$ at p is strictly between π and 3π, then the turning angle of $\tau(s,t)$ at p is strictly between $-\pi$ and π. In this case, $\tau(s,t)$ cannot be the shortest path from s to t in this homotopy class, so $\text{len}(\sigma_{h'}(s,t)) < \text{len}(\tau(s,t)) = \text{len}(\sigma_h(s,t))$.

On the other hand, if the turning angle of $\sigma_h(s,t)$ at p is at least 3π, then the turning angle of $\tau(s,t)$ at p is at least π, which implies that $\tau(s,t)$ is the shortest path from s to t in h'. In this case $\sigma_h(s,t)$ and $\sigma_{h'}(s,t) = \tau(s,t)$ are geometrically equivalent and thus have equal length, but $\text{turn}(\sigma_{h'}(s,t)) = \text{turn}(\sigma_h(s,t)) - 2\pi < \text{turn}(\sigma_h(s,t))$.

Hence $\sigma_{h'}(s,t) \prec \sigma_h(s,t)$ for all s and t, which contradicts our assumption that h is a minimal homotopy class. We conclude that no lifted obstacle point $\hat{p}$ lies on every shortest path $\hat{\sigma}_h(s,t)$.

STAGE 2: HOMOTOPY CONSTRUCTION. If the shortest path $\hat{\sigma}_h(0,0)$ is a proper line segment, then the geodesic $\sigma_h(0,0)$ is also a proper line segment, and the proof is complete. Thus, we assume that $\hat{\sigma}_h(0,0)$ passes through at least one point in $\hat{P}$.

Let $\hat{p}_1, \ldots, \hat{p}_k$ be the sequence of lifted obstacle points on the shortest path $\hat{\sigma}_h(0,0)$. (The points $\hat{p}_i$ are distinct, although their projections back into $\mathcal{E}$ might not be.) Our previous argument implies that for each i, there is a pair of parameters (s_i, t_i) such that $\hat{\sigma}_h(s_i, t_i)$ does not pass through $\hat{p}_i$.

We consider a continuous motion of the parameter point (s,t), starting at $(s,t) = (0,0)$ and then moving successively to each point (s_i, t_i). Specifically, we define two continuous functions $s\colon [0,k] \to [0,m]$ and $t\colon [0,k] \to [0,n]$ such that $s(0) = t(0) = 0$, and for any integer i, we have $s(i) = s_i$ and $t(i) = t_i$. To simplify our notation, we write $\hat{\sigma}(\tau)$ to denote the shortest path $\hat{\sigma}_h(s(\tau), t(\tau))$.

As the parameter τ ('time') increases, points in $\hat{P}$ are inserted into and deleted from the deque of vertices of $\hat{\sigma}(\tau)$. If the deque is empty at any time τ, then the shortest path $\hat{\sigma}(\tau)$ is a proper line segment, which implies that the projected path $\sigma(\tau)$ is a proper line segment as well, concluding the proof. Thus, we assume to the contrary that the deque is never empty. Each vertex $\hat{p}_1, \ldots, \hat{p}_k$ must be deleted from the deque at some time during the motion (but may be reinserted later).

Suppose $\hat{p}$ is the *last* point among $\hat{p}_1, \ldots, \hat{p}_k$ to be removed from the deque for the *first* time. Without loss of generality, we assume $\hat{p}$ is first removed from the front of the deque at time τ_1. Let $\hat{q}$ denote the second point in the deque just before $\hat{p}$ is removed; this point must exist, because the deque is never empty. The point $\hat{p}$ lies on the first segment $\hat{a}\hat{q}$ of $\hat{\sigma}(\tau_1)$, where $\hat{a} = \hat{A}(s(\tau_1))$.

By definition of $\hat{p}$, point $\hat{q}$ must have been pushed onto the *back* of in the deque at some earlier time $\tau_2 < \tau_1$. Just after $\hat{q}$ is inserted, the last two points in the deque are $\hat{p}$ and $\hat{q}$, in that order. Moreover, $\hat{q}$ lies on the last segment $\hat{p}\hat{b}$ of $\hat{\sigma}(\tau_2)$, where $\hat{b} = \hat{B}(t(\tau_2))$.

Thus, there is an *improper* line segment $\hat{a}\hat{b}$ between a point in $\hat{A}$ and a point in $\hat{B}$. Since all line segments in $\hat{\mathcal{E}}$ are shortest paths, $\hat{a}\hat{b}$ is the shortest path $\hat{\sigma}_h(\tau_1, \tau_2)$. Thus, the path $\sigma_h(\tau_1, \tau_2)$ in $\mathcal{E}$ is an *improper* line segment in relative homotopy class h. Finally, for sufficiently small $\varepsilon > 0$, one of the four paths $\sigma_h(\tau_1 \pm \varepsilon, \tau_2 \pm \varepsilon)$ is a proper line segment (because no three vertices of the input

are collinear). □

Proposition 5.3.3 implies that we can enumerate the set of minimal relative homotopy classes in polynomial time as follows. Call a line segment ab with $a \in A$ and $b \in B$ *extremal* if it satisfies one of the following conditions:

(1) The endpoints of ab are vertices of A and B.

(2) One endpoint is a vertex of A or B and the segment contains one point in P.

(3) The segment contains two points in P.

Every proper line segment is relatively homotopic to at least one extremal line segment. Conversely, every extremal line segment carries paths in at most four minimal relative homotopy classes, distinguished by assigning turning angles of π or $-\pi$ at the obstacle vertices that lie on the segment. Thus, to enumerate the minimal relative homotopy classes, it suffices to enumerate the extremal line segments.

There are $O(mn)$ extremal segments of type (1), which we can easily enumerate in $O(mn)$ time by brute force. Each vertex $a \in A$ and point $p \in P$ determine at most n extremal segments of type (2), one for each intersection between $\overrightarrow{ap}$ and B. Similarly, each vertex $b \in B$ and point $p \in P$ determine at most n extremal segments of type (2). Thus, there are $O(mnk)$ extremal segments of type (2); again, we can easily enumerate these in $O(mnk)$ time. Finally, any two points $p, q \in P$ determine $O(mn)$ extremal segments of type (3), distinguished by the intersection points of $\overleftrightarrow{pq}$ with A and B, so there are $O(mnk^2)$ type-(3) extremal segments in total. For any obstacle points p and q, we can compute the intersection points $\overleftrightarrow{pq} \cap A$ and $\overleftrightarrow{pq} \cap B$ in $O(m+n)$ time, and then enumerate the extremal segments that contain both p and q in $O(mn)$ time, again by brute force.

Altogether, we enumerate $O(mnk^2)$ extremal line segments, and therefore $O(mnk^2)$ minimal homotopy classes, in $O(mnk^2)$ time.

There are polygonal curves and point sets that admit $\Omega(mnk^2)$ distinct minimal relative homotopy classes; see Figure 5.2 for an example. Thus, any improvement in this portion of the algorithm will require a finer characterization of optimal relative homotopy classes.

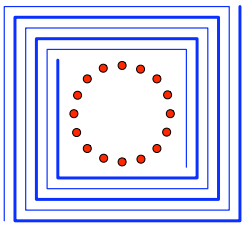

Figure 5.2. An input with $\Omega(N^4)$ minimal relative homotopy classes.

5.3.3 Polygonal Obstacles

The preceding argument breaks down if we allow polygonal obstacles; indeed, it is easy to construct instances where there are *no* proper line segments. Even if the instance allows proper line segments, the optimal homotopy class may not include one. Thus, we require a more complex characterization of minimal homotopy classes in this case. Note that we allow obstacles to be arbitrary polygonal objects, so the curves A and B may themselves be obstacles also.

We say that obstacle vertices p and q *pin* a geodesic γ if the globally shortest path from p to q (in the closure of $\mathcal{E}$) is a subpath of γ; note that p and q may be the same point. A relative homotopy class h is *pinned* if some pair of obstacle vertices pins every geodesic in h. The intersection of all geodesics within a pinned homotopy class is a shortest path between obstacle vertices, which we call the *pinned subpath* of h.

Lemma 5.3.4. *Every minimal relative homotopy class either contains a proper line segment or is pinned.*

Proof: As in the proof of Proposition 5.3.3, let h be an arbitrary minimal homotopy class. Let $\hat{A}$ and $\hat{B}$ be lifts of A and B in the universal cover $\hat{\mathcal{E}}$, such that for all s and t, the shortest path $\hat{\sigma}_h(s,t)$ between $\hat{A}(s)$ and $\hat{B}(t)$ is a lift of $\sigma_h(s,t)$. Let $\hat{P}$ denote the set of all lifts of the *vertices* of obstacles in P; again, every point in $\hat{P}$ lies on the boundary of $\hat{\mathcal{E}}$.

Let $\hat{\pi}_h$ denote the intersection of all shortest paths $\hat{\sigma}_h(s,t)$. If $\hat{\pi} = \varnothing$, then by Stage 2 in the proof of Proposition 5.3.3, h contains a proper line segment.

Otherwise, $\hat{\pi}_h$ is a shortest path between some pair of lifted obstacle vertices $\hat{p}$ and $\hat{q}$. (In the special case where $\hat{\pi}_h$ is a single point, we have $\hat{p} = \hat{q} = \hat{\pi}_h$.) Now $\hat{p}$ and $\hat{q}$ are lifts of obstacle vertices p and q (which may be the same point, even if $\hat{p}$ and $\hat{q}$ are not), and $\hat{\pi}_h$ is similarly a lift of some path π_h from p to q.

Let σ_h denote the globally shortest path from p to q, and suppose that $\pi_h \neq \sigma_h$. For each s and t, let $\tau(s,t)$ be the curve obtained from $\sigma_h(s,t)$ by replacing π_h with σ_h. All paths $\tau(s,t)$ belong to the same relative homotopy class, which we will denote h'. We now easily confirm that $h' \prec h$, contradicting our assumption that h is minimal. We conclude that π_h is the shortest path from p to q, which implies that h is a pinned relative homotopy class, and π_h is its pinned subpath. □

We call a geodesic γ *direct* if it consists of a proper line segment from A to some obstacle vertex p, the shortest path from p to an obstacle vertex q, and a proper line segment from q to B.

Lemma 5.3.5. *Every pinned relative homotopy class contains a direct geodesic.*

Proof: Let h be a pinned homotopy class, and let p and q denote the start and end of the pinned subpath π_h. Any geodesic $\sigma_h(s,t)$ consists of a geodesic $\alpha_h(s)$ from A to p, the pinned subpath π_h, and a geodesic $\beta_h(t)$ from q to B.

If $\alpha_h(0)$ is a proper segment, our claim is proved. Thus, we assume that $\alpha_h(0)$ is not a direct geodesic, which implies that the lifted path $\hat{\alpha}_h(0)$ passes through at least one obstacle vertex other than its endpoint $\hat{p}$. Let $\hat{p}^-$ be the last lifted obstacle vertex on $\hat{\alpha}_h(0)$ before $\hat{p}$. Let s_0 be the largest value such that $\hat{\alpha}_h(s)$ contains $\hat{p}^-$ for all $0 \leq s \leq s_0$. Because $\hat{p}^-$ is not on the pinned subpath, it is not on every geodesic $\hat{\alpha}_h(s)$, which implies that $s_0 < m$. For sufficiently small $\varepsilon > 0$, the geodesic $\alpha_h(s_0 + \varepsilon)$ is a proper line segment.

A similar argument implies that $\beta_h(t)$ is a proper line segment for some t. □

We can compute all homotopy classes of proper line segments using only a small modification of our earlier algorithm. In fact, we begin by running the earlier enumeration algorithm using only the vertices of polygons in P as obstacles. We then discard any line segment that intersects the interior of any obstacle polygon.

Lemma 5.3.5 implies that we can enumerate a superset of of the minimal relative homotopy classes in polynomial time. We begin by computing the shortest paths between every pair of obstacle vertices [78]. Next, for every pair p and q of obstacle vertices, we extend the ray $\overrightarrow{pq}$ until it reaches the interior of an obstacle (or infinity), and then compute all $O(m + n)$ intersections between the resulting line segment and the curves A and B. Finally, we concatenate all $O(mk)$ initial segments, $O(k^2)$ shortest paths, and $O(nk)$ final segments to obtain $O(mnk^4)$ pinned paths in $O(mnk^4) = O(N^6)$ time. Every pinned relative homotopy class contains one of these paths.

Unlike Proposition 5.3.3, we do not have an exact characterization of minimal homotopy classes for polygonal obstacles—direct geodesics do not necessarily lie in pinned homotopy classes, and not all pinned homotopy classes are minimal. However, there are problem instances with $\Omega(mnk^4)$ pinned homotopy classes; for example, replace each obstacle point in Figure 2 with a small triangle.

5.3.4 Non-Polygonal Obstacles

We can further generalize our characterization of minimal homotopy classes to arbitrary non-polygonal obstacles. If we replace 'obstacle vertex' with 'obstacle boundary point' in the definitions, Lemmas 5.3.4 and 5.3.5 are still true in this more general setting. In this section, we redefine the necessary terms and generalize the necessary proofs.

Again, we say that two obstacle boundary points p and q *pin* a geodesic γ if the globally shortest path from p to q (in the closure of $\mathcal{E}$) is a subpath of γ; note that p and q may be the same point. A relative homotopy class h is *pinned* if some pair of obstacle boundary points pins every geodesic in h. The intersection of all geodesics within a pinned homotopy class is a shortest path between obstacle boundary points, which we call the *pinned subpath* of h.

Let $\mathcal{O}$ be the set of obstacles present in the space $\mathcal{E}$.

Lemma 5.3.6. *Every minimal relative homotopy class either contains a proper line segment or is pinned.*

Proof: As in the proof of Proposition 5.3.3, let h be an arbitrary minimal homotopy class. Let $\hat{A}$ and $\hat{B}$ be lifts of A and B in the universal cover $\hat{\mathcal{E}}$, such that for all s and t, the shortest path $\hat{\sigma}_h(s,t)$ between $\hat{A}(s)$ and $\hat{B}(t)$ is a lift of $\sigma_h(s,t)$. Let $\hat{\mathcal{O}}$ be the set of all lifts of obstacles $\mathcal{O}$.

Let $\hat{\pi}_h$ denote the intersection of all shortest paths $\hat{\sigma}_h(s,t)$. If $\hat{\pi} = \varnothing$, then by Stage 2 in the proof of Proposition 5.3.3, h contains a proper line segment.

Otherwise, $\hat{\pi}_h$ is a shortest path between some pair of lifted obstacle boundary points, which will we denote by $\hat{p}$ and $\hat{q}$. (In the special case where $\hat{\pi}_h$ is a single point, we have $\hat{p} = \hat{q} = \hat{\pi}_h$.) Now $\hat{p}$ and $\hat{q}$ are lifts of obstacle boundary points p and q (which may be the same point, even if $\hat{p}$ and $\hat{q}$ are not), and $\hat{\pi}_h$ is similarly a lift of some path π_h from p to q.

Let σ_h denote the globally shortest path from p to q, and suppose that $\pi_h \neq \sigma_h$. For each s and t, let $\tau(s,t)$ be the curve obtained from $\sigma_h(s,t)$ by replacing π_h with σ_h. All paths $\tau(s,t)$ belong to the same relative homotopy class, which we will denote h'. We now easily confirm that $h' \prec h$, contradicting our assumption that h is minimal. We conclude that π_h is the shortest path from p to q, which implies that h is a pinned relative homotopy class, and π_h is its pinned subpath. □

We call a geodesic γ *direct* if it consists of a line segment from A to some obstacle boundary point p which is tangent to the obstacle at p, the shortest path from p to an obstacle boundary point q, and a line segment from q to B which is a tangent to the obstacle at q.

Lemma 5.3.7. *Every pinned relative homotopy class contains a direct geodesic.*

Proof: Let h be a pinned homotopy class, and let p and q denote the start and end of the pinned subpath π_h. Any geodesic $\sigma_h(s,t)$ consists of a geodesic $\alpha_h(s)$ from A to p, the pinned subpath π_h, and a geodesic $\beta_h(t)$ from q to B.

If $\alpha_h(0)$ is a proper segment, our claim is proved. Thus, we assume that $\alpha_h(0)$ is not a direct geodesic, which implies that the lifted path $\hat{\alpha}_h(0)$ passes through at least one obstacle boundary point other than its endpoint $\hat{p}$. Let $\hat{p}^-$ be the last lifted obstacle boundary point on $\hat{\alpha}_h(0)$ before $\hat{p}$. Let s_0 be the largest value such that $\hat{\alpha}_h(s)$ contains $\hat{p}^-$ for all $0 \leq s \leq s_0$. Because $\hat{p}^-$ is not on the pinned subpath, it is not on every geodesic $\hat{\alpha}_h(s)$, which implies that $s_0 < m$. For sufficiently small $\varepsilon > 0$, the geodesic $\alpha_h(s_0 + \varepsilon)$ is a proper line segment.

A similar argument implies that $\beta_h(t)$ is a proper line segment for some t. □

If we have a set of obstacles for which we can compute tangents in polynomial time, Lemma 5.3.7 implies that we can enumerate a superset of of the

minimal relative homotopy classes in polynomial time. We begin by computing tangent curves between every pair of obstacles. For each pair of obstacles, we use these precomputed tangent curves to find the shortest path connecting the two obstacles. Next, for each obstacle and line segment in A and B, the set of tangents from points on the curve to the obstacle is a pair of continuous functions (since each point on the curve has exactly two tangents to any obstacle). Assuming the obstacles are semi-algebraic sets defined by functions of bounded degree, we can compute all tangent curves from a line segment to an obstacle which do not pass through another obstacle [101, 102].

Finally, to enumerate the set of direct geodesics, we pick every possible pair of obstacles and enumerate all direct geodesics which contain the shortest path between boundary points on the two obstacles as a pinned subpath. The time to compute direct geodesics depends on the exact obstacles given.

As in the previous section, we do not have an exact characterization of minimal homotopy classes for non-polygonal obstacles —direct geodesics do not necessarily lie in pinned homotopy classes, and not all pinned homotopy classes are minimal.

We will not consider non-polygonal obstacles in our later algorithmic results. Assuming tangents and visibility complexes can be computed, the same algorithms will work, but running times are dependant on the exact type of obstacles present.

5.4 Fréchet Distance in One Homotopy Class

In this section, we describe an algorithm to compute the Fréchet distance $\mathcal{F}_h(A, B)$ in some relative homotopy class h. Our algorithm is a direct adaptation of Alt and Godau's algorithm for computing the classical Fréchet distance between polygonal paths in the plane [3].

As in the previous section, for any $s \in [0, m]$ and $t \in [0, n]$, let $\sigma_h(s, t)$ denote the shortest path from $A(s)$ to $B(t)$ in homotopy class h, and let $d_h(s, t) = \text{len}(\sigma_h(s, t))$. For any $\varepsilon > 0$, let $F_\varepsilon \subseteq [0, m] \times [0, n]$ denote the *free space* $\{(s, t) \mid d_h(s, t) \leq \varepsilon\}$. Our goal is to compute the smallest value of ε such that F_ε contains a monotone path from $(0, 0)$ to (m, n); this is precisely the Fréchet distance $\mathcal{F}_h(A, B)$.

The parameter space $[0, m] \times [0, n]$ decomposes naturally into an $m \times n$ grid; let $C_{i,j} = [i - 1, i] \times [j - 1, j]$ denote the grid cell representing paths from the ith edge of A to the jth edge of B. Our generalization of Alt and Godau's algorithm requires that the restriction of the function d_h to any grid cell $C_{i,j}$ is convex. We prove this fact in Section 5.4.1 (Proposition 5.4.3).

As input to our problem, we are given a path $\sigma_h(s_0, t_0)$ in relative homotopy class h; based on the results of the previous section, this is either a proper line segment or a direct geodesic. Without loss of generality, we assume that the endpoints $A(s_0)$ and $B(t_0)$ are vertices of A and B; otherwise, we insert them

as new vertices and reparameterize.

5.4.1 Geodesic Distance Is Convex

Let $A, B\colon [0,1] \to \mathcal{E}$ be linear motions in the plane with obstacles, with (constant) derivatives $\vec{a}$ and $\vec{b}$ respectively, and let h be a relative homotopy class. For each $t \in [0,1]$, let $\sigma(t)$ be the shortest path from $A(t)$ to $B(t)$ in relative homotopy class h, and let $d(t)$ be the length of $\sigma(t)$. The goal of this section is to prove that the function d is convex.

We omit the easy proof of the following lemma.

Lemma 5.4.1. *Let o be a fixed point in the plane, and let $\varphi\colon \mathbb{R}^2 \to \mathbb{R}$ be the function $p \mapsto \|\overrightarrow{op}\|$. The function φ is convex everywhere, and of class C^1 everywhere except at o. The gradient of φ at any point $p \neq o$ is $\overrightarrow{op}/\|\overrightarrow{op}\|$.*

Fix $t \in [0,1]$. The shortest path $\sigma(t)$ is a polygonal line. Let $\vec{u}(t)$ be the unit vector representing the direction of the first line segment of $\sigma(t)$ (at its initial point $A(t)$). Similarly, we denote by $\vec{v}(t)$ the unit vector representing the direction of the last line segment of $\sigma(t)$.

Recall that, as t increases, the shortest path $\sigma(t)$ encounters a finite number of *events*, between which the set of vertices of the obstacles at which it bends is the same.

Lemma 5.4.2. *Between any two consecutive events, d is convex and of class C^1. In particular, $d'(t) = \vec{b} \cdot \vec{v}(t) - \vec{a} \cdot \vec{u}(t)$, where $\cdot$ denotes inner product.*

Proof: Fix two consecutive events t_0 and t_1.

Assume first that for all t between t_0 and t_1, the path $\sigma(t)$ is *not* a line segment. Then for every $t \in [t_0, t_1]$, $\sigma(t)$ is the concatenation of a line segment from $A(t)$ to a fixed obstacle vertex p, a geodesic from p to another fixed obstacle vertex q, and the line segment from q to $B(t)$. It follows that $d(t)$ equals a constant plus $\|\overrightarrow{pA(t)}\| + \|\overrightarrow{qB(t)}\|$. Our result is now a consequence of Lemma 5.4.1. Specifically, d is the sum of two convex functions, and is therefore convex. Since A and B do not meet the obstacle vertices M and N, the function d is C^1 in the interval $[t_0, t_1]$. The chain rule implies the claimed expression for d'.

If $\sigma(t)$ is a line segment whenever $t_0 \leq t \leq t_1$, then $d(t) = \|\overrightarrow{A(t)B(t)}\|$. Since the function $t \mapsto \overrightarrow{A(t)B(t)}$ is affine, Lemma 5.4.1 also implies that d is convex and of class C^1, and that

$$d'(t) = (\vec{b} - \vec{a}) \cdot \overrightarrow{A(t)B(t)}/\|\overrightarrow{A(t)B(t)}\|.$$

Finally, we observe that

$$\vec{u}(t) = \vec{v}(t) = \overrightarrow{A(t)B(t)} \,/\, \|\overrightarrow{A(t)B(t)}\|,$$

which completes the proof. □

Proposition 5.4.3. *The function d is convex.*

Proof: Lemma 5.4.2 implies that consecutive events, the function d' is continuous and non-decreasing. Let t_0 be an arbitrary event. Since the functions $t \mapsto \vec{u}(t)$ and $t \mapsto \vec{v}(t)$ are continuous at t_0, Lemma 5.4.2 implies that d' is also continuous at t_0. Thus, d' is non-decreasing over the entire interval $[0, 1]$, which implies that d is convex. □

Proposition 5.4.3 implies that the *bivariate* shortest-path distance function $D(u, v)$ between $A(u)$ and $B(v)$ is also convex, as follows. For any $u, u', v, v', t \in [0, 1]$ we denote by $d_{u,v,u',v'}(t)$ the shortest-path distance between $A((1-t)u + tu')$ and $B((1-t)v + tv')$. In other words, we define

$$d_{u,v,u',v'}(t) = D((1-t)(u, v) + t(u', v')).$$

Proposition 5.4.3 implies that the univariate function $d_{u,v,u',v'}$ is convex. It follows that

$$\begin{aligned} D((1-t)(u, v) + t(u', v')) &= d_{u,v,u',v'}((1-t) \cdot 0 + t \cdot 1) \\ &\leq (1-t)d_{u,v,u',v'}(0) + td_{u,v,u',v'}(1) \\ &= (1-t)D(u, v) + tD(u', v'), \end{aligned}$$

which expresses the convexity of D.

The convexity of D immediately implies the following corollary.

Corollary 5.4.4. *For all integers i and j, the restriction of d_h to any grid cell $C_{i,j}$ is convex.*

5.4.2 Preprocessing for Distance Queries

The only significant difference between our algorithm and Alt and Godau's is that we require additional preprocessing to compute several *critical distances* and an auxiliary data structure to answer certain *distance queries*. (If there are no obstacles, each critical distance can be computed, and each distance query can be can be answered, in constant time.)

There are three types of critical distances:

- *Endpoint distances* $d_h(0, 0)$ and $d_h(m, n)$,
- *Vertex-edge distances* $d_h(i, [j-1, j]) = \min\{d_h(i, t) \mid t \in [j-1, j]\}$ for all integers $i \in [0, m]$ and $j \in [1, n]$, and
- *Edge-vertex distances* $d_h([i-1, i], j) = \min\{d_h(s, j) \mid s \in [i-1, i]\}$ for all integers $i \in [1, m]$ and $j \in [0, n]$.

Given integers i and j and any real value ε, a *horizontal distance query* asks for all values of $t \in [j-1, j]$ such that $d_h(i, t) = \varepsilon$, and a *vertical distance query*

asks for all values of $s \in [i-1, i]$ such that $d_h(s, j) = \varepsilon$. The convexity of d_h within any grid cell implies that any distance query returns at most two values.

We first describe how to preprocess a single vertical edge in the parameter grid to answer distance queries; computation of the critical values will be done automatically during the preprocessing. Obviously a similar result applies to horizontal grid edges.

Lemma 5.4.5. *Suppose we are given a point p and a line segment $\ell = \overline{xy}$, parameterized over $[0, 1]$, as well as the geodesic $\sigma_h(p, x)$ and its length $d_h(p, x)$. In $O(k \log k)$ time, we can build a data structure of size $O(k)$ such that for any ε, all values $t \in [0, 1]$ such that $d_h(p, \ell(t)) = \varepsilon$ can be computed in $O(\log k)$ time. We also report the critical vertex-edge distance $d_h(p, \ell)$, the path $\sigma_h(p, y)$, and its length $d_h(p, y)$.*

Proof: We first compute a constrained Delaunay triangulation of the polygons P, the segment ℓ, and point p in time $O(k \log k)$. This triangulation includes ℓ and the edges of polygons in P as edges.

We apply the following observations used in the *funnel* algorithm for computing shortest homotopic paths [28, 87, 77]. The shortest homotopic paths $\sigma_h(p, x)$ and $\sigma_h(p, y)$ may share a common subpath and then split at some vertex v; this vertex is then the apex of two concave chains that form a funnel with base xy. Each concave chain has complexity at most k and intersects a given edge of the triangulation at most twice.

The geodesic from p to x may have complexity greater than $O(k)$, but (as observed above) the concave chain from v to x will have at most $O(k)$ segments. Our goal is to find a vertex w on $\sigma_h(p, x)$ such that the path from w to x contains v. In other words, the chain from w to x along $\sigma_h(p, x)$ will be of complexity $O(k)$ and will contain the concave funnel path.

To find w, walk along the path from x to p. If we find a vertex where the chain is not concave, we must have passed v, so we mark the non-concave vertex as w. If we ever re-cross a segment of the triangulation a second time, we again must have passed the funnel apex v so we can mark the second crossing as w. (We walked along $O(k)$ edges of the chain to find w.) Let π be the portion of $\sigma_h(p, x)$ between p and w, and τ_1 be the portion of $\sigma_h(p, x)$ between w and x.

We know that π is contained in $\sigma_h(p, y)$, since w is before the apex of the funnel v. Let τ_2 be the portion of $\sigma_h(p, y)$ between w and y; this can be computed in $O(k)$ time using the funnel algorithm. Given τ_2, we can then find the apex of the funnel v in $O(k)$ time.

Imagine extending each line segment on the concave chains until it intersects ℓ, the line connecting x and y. Between the two concave chains, the combinatorial description of the distance function changes only at points where the extended lines meet ℓ. To answer distance queries, we will record the $O(k)$ intersections of the extended lines with ℓ. For each of the resulting intervals, record the (fixed) length of the geodesic up to the first vertex in the extended

line, as well as the equations of the two lines that bracket the interval. In constant time per interval, we can also compute and store the value $t^* \in [0,1]$ such that $d_h(p, \ell(t^*))$ is minimized, along with the path $\sigma_h(p, \ell(t^*))$; this gives the desired value $d_h(p, \ell)$.

The funnel data structure requires $O(k)$ space to store the $O(k)$ combinatorial changes to the leash as its endpoint sweeps $\ell = \overline{xy}$.

Now given this data structure, we answer distance queries as follows. If the distance queried is smaller than $d_h(p, \ell)$, we return the empty set. If it is equal to $d_h(p, \ell)$, we return $\ell(t^*)$. If it is larger than $d_h(p, \ell)$, we do two binary searches, one on the intervals between x and $\ell(t^*)$ and the other on the intervals between $\ell(t^*)$ and y. □

Lemma 5.4.6. *In $O(mnk \log k)$ time and using $O(mnk)$ space, we can compute all critical distances, as well as a data structure of size $O(mnk)$ that can answer any horizontal or vertical distance query in $O(\log k)$ time.*

Proof: We preprocess each edge of the parameter grid as described in the previous lemma, Lemma 5.4.5. We start from the vertex (i, j) that is our given input, either a straight line segment or a direct geodesic. We then walk on the edges of the grid, visiting each edge at least once and at most $O(1)$ times. During this walk, at each current vertex (i, j), we maintain the shortest homotopic path $\sigma_h(i, j)$ and its length $d_h(i, j)$. Each time we walk along an edge, we apply Lemma 5.4.5 to preprocess it and to compute the shortest homotopic path corresponding to the target vertex of that edge. Each step takes $O(k \log k)$ time, and there are $O(mn)$ edges, whence the running-time.

As we walk along an edge of the parameter grid, we use a deque to push and pop the obstacle vertices along the leash in constant time per operation. Since at most k vertices are pushed onto the deque for each grid edge, the total size of the deque is $O(mnk)$. □

5.4.3 Decision Procedure

Like Alt and Godau, we first consider the following *decision problem*: Is $\mathcal{F}_h(A, B)$ at least some given value ε? Equivalently, is there a monotone path in the free space F_ε from $(0, 0)$ to (m, n)? Our algorithm to solve this decision problem is identical to Alt and Godau's, except for the $O(\log k)$-factor penalty for distance queries; we briefly sketch it here for completeness.

For any integers i and j, let $h_{i,j}$ denote the intersection of the free space F_ε with the horizontal edge $([i-1, i], j)$, and let $v_{i,j}$ denote the intersection of F_ε with the vertical edge $(i, [j-1, j])$. In the first phase of the decision procedure, we compute $h_{i,j}$ and $v_{i,j}$ for all i and j, using one distance query (and $O(\log k)$ time) for each edge of the parameter grid.

In the second phase of the decision procedure, we propagate in lexicographic order from $C_{1,1}$ to $C_{m,n}$ and determine which $h_{i,j}$ and $v_{i,j}$ are reachable via a

monotone path from $C_{1,1}$. Since the free space in each $C_{i,j}$ is convex, we can propagate through each cell in constant time.

Our decision algorithm returns true if and only if there is a monotone path that reaches (m, n). The total running time is $O(mn \log k)$.

5.4.4 Computing Fréchet Distance

Finally, we describe how to use our decision procedure to compute the optimum value $\varepsilon^* = \min\{\varepsilon \mid (m, n) \in R_\varepsilon\}$; this is the Fréchet distance $\mathcal{F}_h(A, B)$.

We start by computing critical distances and the distance-query data structure in $O(mnk \log k)$ time, as described in Lemma 5.4.6. We then sort the $O(mn)$ critical distances. Using the decision procedure, we can compare the optimal distance ε^* with any critical distance ε in $O(mn \log k)$ time. By binary search, we can, repeating this step $O(\log mn)$ times, compute an interval $[\varepsilon^-, \varepsilon^+]$ that contains ε^* but no critical distances.

We then apply Megiddo's *parametric search* technique [91]; see also [31, 119]. Parametric search combines our decision procedure with a 'generic' parallel algorithm whose combinatorial behavior changes at the optimum value ε^*. Alt and Godau observe that one of two events occurs when $\varepsilon = \varepsilon^*$:

- For some integers i, i', j, the bottom endpoint of $v_{i,j}$ and the top endpoint of $v_{i',j}$ lie on the same horizontal line.
- For some integers i, j, j', the left endpoint of $h_{i,j}$ and the right endpoint of $h_{i,j'}$ lie on the same vertical line.

Thus, it suffices to use a 'generic' algorithm that sorts the $O(mn)$ endpoint *values* of all non-empty segments $h_{i,j}$ and $v_{i,j}$, where the value of an endpoint (s, j) of $h_{i,j}$ is s, and the value of an endpoint (i, t) of $v_{i,j}$ is t.

We use Cole's parallel sorting algorithm [32], which runs in $O(\log N)$ parallel steps on $O(N)$ processors, as our generic algorithm. Each parallel step of Cole's sorting algorithm needs to compare $O(mn)$ endpoints. The graph of an endpoint, considered as a function of ε, is convex, monotone, and made of $O(k)$ pieces, each having a simple closed form (see proof of Lemma 5.4.5). It follows that the sign of a comparison between two endpoints may change at $O(k)$ different values of ε that can be computed in $O(k)$ time. Applying the parametric search paradigm requires the following operations for each parallel step of the sorting algorithm:

- Compute the $O(mnk)$ values of ε corresponding to the changes of sign of the $O(mn)$ comparisons. This can be done in $O(mnk)$ time and $O(mnk)$ space.
- Apply binary search to these values by median finding, calling the decision procedure to discard half of them at each step of the search. This takes $O(mnk + T_d \log(mnk))$ time, where $T_d = O(mn \log k)$ is the running time

of our decision procedure. We obtain this way an interval for ε where each of the $O(mn)$ comparisons has a determined sign.

- Deduce in $O(mn \log k)$ time the sign of each of the $O(mn)$ comparisons within the previously computed interval.

Since the underlying sorting algorithm requires $O(\log mn)$ parallel steps, the resulting parametric search algorithm runs in time:

$$O(mn \log(mn)(k + \log k \log(mnk)) = O(N^3 \log N)$$

The distance query data structure requires $O(mnk)$ space. We require an additional $O(mnk)$ space to simulate sequentially each parallel step of the sorting algorithm; we can re-use this space for subsequent parallel steps. Therefore, the total space complexity of our algorithm is $O(mnk) = O(N^3)$.

Lemma 5.4.7. *Given either a proper line segment or a direct geodesic any minimal relative homotopy class h, the Fréchet distance $\mathcal{F}_h(A, B)$ can be computed in $O(N^3 \log N)$ time and $O(N^3)$ space.*

5.4.5 Summary

Finally, to compute the homotopic Fréchet distance $\overline{\mathcal{F}}(A, B)$ in the plane minus a set of point obstacles, we compute the Fréchet distance $\mathcal{F}_h(A, B)$ in each of the $O(mnk^2)$ minimal homotopy classes h. Similarly, for polygonal obstacles, we compute $\mathcal{F}_h(A, B)$ for every class h in a set of $O(mnk^4)$ relative homotopy classes, which includes every minimal homotopy class.

Theorem 5.4.8. *The homotopic Fréchet distance between two polygonal curves in the plane minus a set of points can be computed in $O(N^7 \log N)$ time using $O(N^3)$ space.*

Theorem 5.4.9. *The homotopic Fréchet distance between two polygonal curves in the plane minus a set of polygons can be computed in $O(N^9 \log N)$ time using $O(N^3)$ space.*

5.5 Spaces of Non-positive Curvature

It is worth noting that Lemma 5.2.1 will extend to more general spaces. Let $\mathcal{M}$ be a topological space, and consider a triangle T in $\mathcal{M}$ with edges of length a, b, and c. Form a comparison triangle T' in the plane with the same edge lengths. The *CAT(0) inequality* is satisfied in $\mathcal{M}$ if for every choice of a geodesic triangle T, any geodesic chord of T has length less than or equal to the corresponding chord in T'. If this inequality is satisfied, then we say the space is *CAT(0)*.

Any space that admits a metric which is locally CAT(0) is said to have *non-positive curvature*. The relevant fact we rely upon is that any space with

non-positive curvature has a unique geodesic between any two points (see e.g. [14]).

Lemma 5.5.1. *Suppose ℓ is a leash map between two curves A and B in $\mathcal{M}$. There is a geodesic leash map ℓ' between A and B such that for all $t \in [0,1]$, the leash $\ell'(t,\cdot)$ is the shortest path homotopic to $\ell(t,\cdot)$ with the same endpoints. Additionally, the length of ℓ' is at most the length of ℓ.*

Proof: The proof proceeds as it did in Lemma 5.2.1. We lift ℓ to the universal cover $\hat{\mathcal{M}}$ of $\mathcal{M}$, obtaining a leash map $\hat{\ell}$ between the lifts $\hat{A}$ and $\hat{B}$ of A and B respectively. For each $t \in [0,1]$, let $\hat{\ell}'(\cdot,t)$ be the shortest path between the endpoints of $\hat{\ell}(t,\cdot)$ in the same homotopy class. The universal cover $\hat{\mathcal{M}}$ is a simply-connected and non-positively curved space, so geodesics are unique. In addition, shortest paths in $\hat{\mathcal{M}}$ vary continuously as the endpoints move continuously. It follows that $\hat{\ell}'$ is a continuous function in both arguments, and therefore a (geodesic) leash map in $\hat{M}$. The projection ℓ' of $\hat{\ell}'$ back to $\mathcal{M}$ is a (geodesic) leash map between A and B. For all t, the leash $\ell'(t,\cdot)$ is the shortest path homotopic to $\ell(t,\cdot)$, so $\max_t \operatorname{len}(\ell'(t,\cdot)) \leq \max_t \operatorname{len}(\ell(t,\cdot))$.

□

This means that there is hope of generalizing homotopic Fréchet distance to more general surfaces. The main question regards the representation of the surface, since some representations of non-positively curved surfaces (such as hyperbolic space) make distance calculations computationally intractable or at least raise issues of numerical approximations. However, if geodesics are computable, then it is possible to compute the homotopic Fréchet distance.

5.6 Open Problems

Improving the running time of our algorithms is the most immediate outstanding open problem. For point obstacles, we conjecture that the running time can be improved by at least a factor of N by optimizing leash maps in every minimal homotopy class simultaneously. Since shortest paths between the same endpoints but belonging to different homotopy classes are related, we expect to (partially) reuse the results of shortest path computations going from one homotopy class to another. For polygonal obstacles, an exact characterization of minimal homotopy classes would almost certainly lead to a significantly faster algorithm.

Cook and Wenk [36] describe an algorithm for computing geodesic Fréchet distance between two curves within a simple polygon, generalizing earlier results of Efrat *et al.* [53]. Their algorithm is faster than ours by roughly a factor of N, in part because they use a randomized strategy in place of parametric search. Unfortunately, we have not been able to apply their technique to our more

general problem, because it seems to require a simply-connected environment. However, similar ideas may simplify and improve our algorithm.

Weak Fréchet distance is a variant of the ordinary Fréchet distance without the requirement that the endpoints move monotonically along their respective curves—the dog and its owner are allowed to backtrack to keep the leash between them short. Alt and Godau [3] gave a simpler algorithm for computing the weak Fréchet distance, using a graph shortest-path algorithm instead of parametric search. A similar simplification of our algorithm computes the weak *homotopic* Fréchet distance between curves in the punctured plane in polynomial time.

With regard to morphing applications [54], it would also be interesting to minimize the average leash length over the course of the homotopy, rather than the maximum leash length.

It would be interesting to compute homotopic Fréchet distance in more general spaces. In particular, we are interested in computing the homotopic Fréchet distance between two curves on a convex polyhedron, generalizing the algorithm of Maheshwari and Yi for classical Fréchet distance [89]. The vertices of the polyhedron are 'mountains' over which the leash can pass only if it is long enough. Shortest paths on the surface of a convex polyhedron do *not* vary continuously as the endpoints move, because of the positive curvature at the vertices, so we cannot consider only geodesic leash maps.

Finally, it would also be interesting to consider the homotopic (weak) Fréchet distance between higher-dimensional manifolds; such problems arise with respect to surfaces in configuration spaces of robot systems. Standard Fréchet distance is difficult to compute in higher dimensions, although the *weak* Fréchet distance between two triangulated surfaces can be computed in polynomial time [2].

Chapter 6

Rips Complexes of Planar Point Sets

6.1 Introduction

The previous chapters have focused on homotopy in combinatorial surfaces and $\mathbb{R}^2$ minus obstacles. Another type of topological space used in various applications is a *simplicial complex*. Homotopy questions in simplicial complexes are generally much more difficult than previous models discussed. For general simplicial complexes, even determining whether two paths are homotopic is undecidable [90]. For this reason, most topological algorithms for simplicial complexes are based on *homology*, which provides a cruder classification of topological features than homotopy, but generalizes more easily to higher dimensions [49, 50, 22, 64, 125].

Given a set X of points in Euclidean space $\mathbb{E}^n$, the *Vietoris-Rips complex* $\mathcal{R}_\epsilon(X)$ is the abstract simplicial complex whose k-simplices are determined by subsets of $k+1$ points in X with diameter at most ϵ. For simplicity, we set $\epsilon = 1$ and write $\mathcal{R} := \mathcal{R}_1(X)$ for the remainder of the paper, with the exception of §6.4. For brevity (and to conform to typical usage), we refer to $\mathcal{R}$ as the *Rips complex*; we denote the 1-skeleton of the Rips complex as $\mathcal{R}^1$. The Rips complex is an example of a *flag complex* — the maximal simplicial complex with a given 1-skeleton.

The Rips complex was used by Vietoris [120] in the early days of homology theory, as a means of creating finite simplicial models of metric spaces. Within the past two decades, the Rips complex has been utilized frequently in geometric group theory [67] as a means of building simplicial models for group actions. Most recently, Rips complexes have been used heavily in computational topology, as a simplicial model for point-cloud data [20, 21, 22, 37] and as simplicial completions of communication links in sensor networks [38, 39, 96].

The utility of Rips complexes in computational topology stems from the ability of a Rips complex to approximate the topology of a cloud of points. We make this notion more specific. To a collection of points, one can assign a different simplicial model called the Čech complex that accurately captures the homotopy type of the cover of these points by balls. Formally, given a set X of points in some Euclidean space $\mathbb{E}^n$, the *Čech complex* $\mathcal{C}_\epsilon(X)$ is the abstract simplicial complex where a subset of $k+1$ points in X determines a k-simplex if

and only if they lie in a ball of radius $\epsilon/2$. The Čech complex is equivalently the nerve of the set of closed balls of radius $\epsilon/2$ centered at points in X. The *Čech theorem* (or *Nerve lemma*, see, *e.g.*, [11]) states that $\mathcal{C}_\epsilon(X)$ has the homotopy type of the union of these balls. Thus, the Čech complex is an appropriate simplicial model for the topology of the point cloud (where the parameter ϵ is a variable).

There is a price for the high topological fidelity of a Čech complex. Given the point set, it is nontrivial to compute and store the simplices of the Čech complex. The virtue of a Rips complex is that it is determined completely by its 1-skeleton — the proximity graph of the points. (This is particularly useful in the setting of *ad hoc* wireless networks, where the hardware establishes communication links based, ideally, on proximity of nodes.) The penalty for this simplicity is that it is not immediately clear what is encoded in the homotopy type of $\mathcal{R}$. Like the Čech complex, it is not generally a subcomplex of its host Euclidean space $\mathbb{E}^n$, and, unlike the Čech complex it need not behave like an n-dimensional space at all: $\mathcal{R}$ may have nontrivial topological invariants (homotopy or homology groups) of dimension n and above.

The disadvantage of both Čech and Rips complexes are in their rigid cut-offs as a function of distance between points. Arbitrarily small perturbations in the locations of the points can have dramatic effects on the topology of the associated simplicial complexes. Researchers in sensor networks are acutely aware of this limitation, given the amount of uncertainty and fluctuation in wireless networks. To account for this, several researchers in sensor networks have used a notion of a distance-based communication graph with a region of uncertain edges [6, 84]. This motivates the following construction.

Fix an open *uncertainty interval* (ϵ, ϵ') which encodes connection errors as a function of distance. For all nodes of distance $\leq \epsilon$, there is an edge, and for all nodes of distance $\geq \epsilon'$, no edge exists. For nodes of distance within (ϵ, ϵ'), a communication link may or may not exist. A *quasi-Rips complex* with uncertainty interval (ϵ, ϵ') is the simplicial flag complex of such a graph. We note that this does not model temporal uncertainty, merely spatial.

A completely different model of simplicial complexes associated to a point cloud comes from considering shadows. Any abstract simplicial complex with vertices indexed by geometric points in $\mathbb{E}^n$ (*e.g.*, a Rips, Čech, or quasi-Rips complex) has a canonical *shadow* in $\mathbb{E}^n$, which strikes a balance between computability and topological faithfulness. For, say, a Rips complex, the canonical *projection* $p\colon \mathcal{R} \to \mathbb{E}^n$ is the well-defined function that maps each simplex in $\mathcal{R}$ affinely onto the convex hull of its vertices in $\mathbb{E}^n$. This projection map is continuous and piecewise-linear. The shadow $\mathcal{S}$ is the image $p(\mathcal{R})$ of this projection map.

We study the topological faithfulness of the projection map p (see Figure 6.1). Specifically, we look at the connectivity of p. Recall that a topological map $f : X \to Y$ is *k-connected* if the induced homomorphisms on homotopy

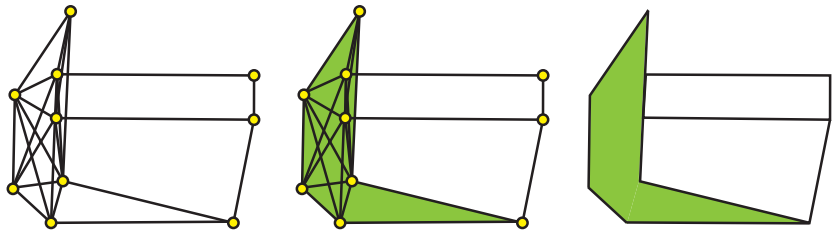

Figure 6.1. A connectivity graph in the plane [left] determines a 5-dimensional (Vietoris-) Rips complex [center] and its 2-dimensional projected shadow [right].

groups $p_* : \pi_i(X) \to \pi_i(Y)$ are isomorphisms for all $0 \leq i \leq k$: *e.g.*, a 1-connected map preserves path-connectivity and fundamental group data.

Our principal results are the following, ordered as they appear in the following sections. For any set of points in $\mathbb{E}^2$, $\pi_1(p)\colon \pi_1(\mathcal{R}) \to \pi_1(\mathcal{S})$ is an isomorphism. As a corollary of this result, we get that the fundamental group of any planar Rips complex is free. Given any finitely presented group G, there exists a quasi-Rips complex $\mathcal{R}_Q$ with arbitrarily small uncertainty interval such that $\pi_1(\mathcal{R}_Q)$ is a free extension of G. The projection map p on $\mathbb{R}^n$ is always k-connected for $k = 0$ or $n = 1$. For all other cases except $(k, n) = (1, 2)$ and, perhaps, $(1, 3)$, k-connectivity fails on $\mathbb{R}^n$ (see Figure 6.8).

6.2 Planar Rips Complexes and Their Shadows

In this section, we restrict our attention to the 2-dimensional case.

6.2.1 The Shadow Complex

The shadow $\mathcal{S}$ is a polyhedral subset of the plane. By Carathéodory's theorem [48], $\mathcal{S}$ is the projection of the 2-skeleton of $\mathcal{R}$. Since the vertices of $\mathcal{R}$ are distinct points in the plane, it follows that distinct edges of $\mathcal{R}$ have distinct images under p, and these are nondegenerate. Informally we will identify vertices and edges of $\mathcal{R}$ with their images under p. On the other hand, p may be degenerate on 2-simplices.

We can canonically decompose $\mathcal{S}$ into a 2-dimensional **shadow complex** as follows:

- A **shadow vertex** is either a vertex of $\mathcal{R}$ or a point of transverse intersection of two edges of $\mathcal{R}$. We write $\mathcal{S}^0$ for the set of shadow vertices.
- A **shadow edge** is the closure of any component of $p(\mathcal{R}^1) \setminus \mathcal{S}^0$. Each shadow edge is a maximal line segment contained in a Rips edge, with no shadow vertices in its interior. We write $\mathcal{S}^1$ for the union of all shadow vertices and shadow edges.
- Finally, a **shadow face** is the closure of any bounded component of $\mathbb{E}^2 \setminus \mathcal{S}^1$.

The fundamental group $\pi_1(\mathcal{S})$ may now be described in terms of combinatorial paths of shadow edges modulo homotopy across shadow faces, whereas $\pi_1(\mathcal{R})$ may be described in terms of combinatorial paths of Rips edges modulo homotopy across Rips faces. This description opens the door to combinatorial methods in the proof that $\pi_1(p)$ is an isomorphism.

6.2.2 Technical Lemmas

Our main result, given in theorem 6.3.1, is that for any planar point set, $\pi_1(\mathcal{R}) \simeq \pi_1(\mathcal{S})$. This result will follow from reduction to three special cases. We prove these cases in this subsection. We use the following notation. Simplices of a Rips complex will be specified by square braces, e.g., $[ABC]$. Images in the shadow complex will be denoted without adornment, e.g., ABC. The Euclidean length of an edge AB will be denoted $|AB|$. Braces $\langle\cdot\rangle$ will be used to denote the span in $\mathcal{R}$: the smallest subcomplex containing a given set of vertices, e.g., $\langle ABCD\rangle$.

The following propositions address the three special cases of Theorem 6.3.1 which are used to prove the theorem. Certain induced subcomplexes of $\mathcal{R}$ are shown to be simply connected. In the first two cases, it is helpful to establish the stronger conclusion that these subcomplexes are **cones**: all maximal simplices share a common vertex, called the **apex**. The first of these cases is trivial and well-known (*viz.*, [38, 60]).

Proposition 6.2.1. *Let $\mathcal{R} = \langle ABYZ\rangle$ be a Rips complex containing simplices $[AB]$ and $[YZ]$ whose images in $\mathcal{S}$ intersect. Then $\mathcal{R}$ is a cone.*

Proof: Let x be the common point of AB and YZ. Each edge is split at x into two pieces, at most one of which can have length more than one-half. The triangle inequality implies that the shortest of these four half-edges must have its endpoint within unit distance of both endpoints of the traversing edge, thus yielding a 2-simplex in $\mathcal{R}$. □

Proposition 6.2.2. *Let $\mathcal{R} = \langle ABXYZ\rangle$ be a Rips complex containing simplices $[AB]$ and $[XYZ]$ whose images in $\mathcal{S}$ intersect. Then $\mathcal{R}$ is a cone.*

Proof: The edge AB intersects the triangle XYZ. If AB intersects only one edge of XYZ, then one vertex of AB (say, A) lies within XYZ and cones off a 3-simplex $[AXYZ]$ in $\mathcal{R}$. Therefore, without a loss of generality we may assume AB crosses ZY and ZX.

By Proposition 6.2.1, the subcomplexes $\langle ABXZ\rangle$ and $\langle ABYZ\rangle$ are cones. If these two cones have the same apex, then the entire Rips complex $\mathcal{R}$ is a cone with that apex. Similarly, if either apex lies inside the image triangle XYZ, then $\mathcal{R}$ is a cone with that apex. The only remaining possibility is that A is the apex of one subcomplex and B is the apex of the other; in this case, $\mathcal{R}$ is a cone over Z, since both A and B are connected to Z. □

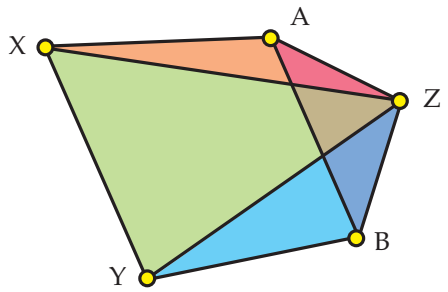

Figure 6.2. The last case of Proposition 6.2.2.

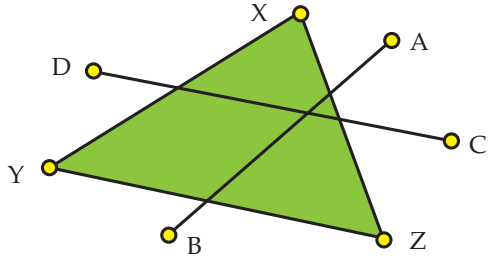

Figure 6.3. The setup for Proposition 6.2.3.

Proposition 6.2.3. *Let $\mathcal{R} = \langle ABCDXYZ \rangle$ be a Rips complex containing simplices $[AB]$, $[CD]$ and $[XYZ]$ whose images in $\mathcal{S}$ meet in a common point. Moreover, assume that none of A, B, C, D lies in the interior of XYZ. Then $\pi_1(\mathcal{R})$ is trivial.*

To prove Proposition 6.2.3, we use two further geometric lemmas.

Lemma 6.2.4. *Let $\mathcal{R} = \langle BXYZ \rangle$ be a Rips complex containing simplex $[XYZ]$. If M is a point in XYZ such that $|BM| \leq \frac{1}{2}$, then $\mathcal{R}$ contains at least one of the edges $[BX]$, $[BY]$, $[BZ]$.*

Proof: If B lies in XYZ then all three edges belong to $\mathcal{R}$. Otherwise, BM meets the boundary of XYZ at a point M'. We may assume without loss of generality that M' lies on XY, with $|M'X| \leq |M'Y|$. Then $|BX| \leq |BM'| + |M'X| \leq \frac{1}{2} + \frac{1}{2} = 1$. □

Lemma 6.2.5. *Let $\mathcal{R} = \langle ABCXYZ \rangle$ be a Rips complex containing simplices $[ABC]$ and $[XYZ]$. Suppose that AB intersects XYZ but BC and AC do not. Then $\mathcal{R}$ is a cone.*

Proof: The hypotheses of the lemma imply that at least one of the points X, Y, or Z lies in the interior of ABC. $\mathcal{R}$ is a cone on this point. □

Proof (Proof of Proposition 6.2.3): We argue by exhaustive case analysis that $\mathcal{R}$ contains no minimal noncontractible cycle.

Suppose γ is a minimal noncontractible cycle in $\mathcal{R}$. Because $\mathcal{R}$ is a flag complex, γ must consist of at least four Rips edges. Our previous Propositions

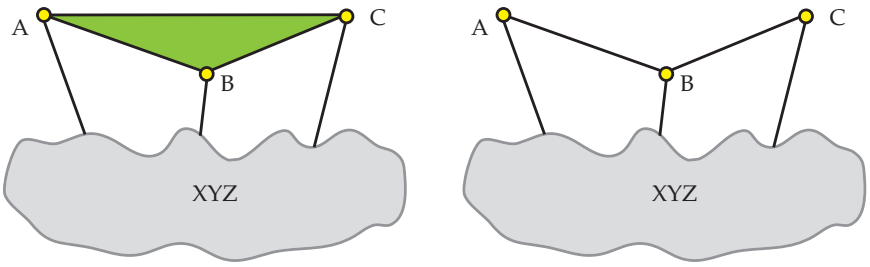

Figure 6.4. $ACXY$ (left), or $ABCX$ (right), splits into two cycles in the presence of $[BX]$, $[BY]$, or $[BZ]$.

imply that this cycle intersects each simplex $[AB]$, $[CD]$, and $[XYZ]$ at least once. By minimality, γ contains at most one edge of $[XYZ]$. Thus, we may assume without loss of generality (by relabeling if necessary) that γ is of the form $A(B)C(D)X(Y)$ where $(\cdot)$ denotes an optional letter; note that at least one of these optional letters must be present, since the cycle must contain four Rips edges.

Claim 1: In a minimal cycle, the subwords $ABCD$, $CDXY$, $XYAB$ are impossible. Proposition 6.2.1 (in the first case) and Proposition 6.2.2 (in the last two cases) imply that the subpaths corresponding to these subwords are homotopic (relative to endpoints) within a cone subcomplex to a path with at most two edges, contradicting the minimality of γ.

Claim 1 implies that that there is at most one optional letter. This leaves three possible minimal noncontractible cycles: $ACXY$, $ABCX$, and $ACDX$. The last two cases differ only by relabeling, so it suffices to consider only $ACXY$ and $ABCX$.

Claim 2: $ACXY$ is impossible. Suppose $ACXY$ is a cycle in $\mathcal{R}$. If AC meets XYZ then Proposition 6.2.2 implies that $\langle ACXYZ\rangle$ is a cone, so $ACXY$ is contractible. Thus, we can assume that AC does not meet XYZ.

By Proposition 6.2.1, either $[BC]$ or $[AD]$ is a Rips edge. Without loss of generality, assume $[BC]$ is a Rips edge; then $[ABC]$ is a Rips triangle. If BC does not meet XYZ, then Lemma 6.2.5 implies that $\langle ABCXYZ\rangle$ is a cone, and hence that $ACXY$ is contractible. Thus we can assume that BC intersects XYZ.

Proposition 6.2.2 now implies that both $\langle ABXYZ\rangle$ and $\langle BCXYZ\rangle$ are cones. If any of the segments $[BX]$, $[BY]$, $[BZ]$ is a Rips edge, then the cycle $ACXY$ is homotopic to the sum of two cycles, contained respectively in the cones $\langle ABXYZ\rangle$ and $\langle BCXYZ\rangle$, and hence is contractible. See Figure 6.4(a).

We can therefore assume that none of the segments $[BX]$, $[BY]$, $[BZ]$ is a Rips edge. In this case, the apex of $\langle ABXYZ\rangle$ must be A. In particular, the diagonal $[AX]$ of the cycle $ACXY$ belongs to $\mathcal{R}$, and so $ACXY$ is contractible. This completes the proof of Claim 2.

Claim 3: $ABCX$ is impossible. Suppose $ABCX$ is a cycle in $\mathcal{R}$. If either

$[AC]$ or $[BX]$ is a Rips edge, then $ABCX$ is trivially contractible. Moreover, if either $[BY]$ or $[BZ]$ is a Rips edge, then the cycle $ABCX$ reduces to the sum of two cycles, as in Figure 6.4(b). The left cycle is contractible by Proposition 6.2.2, and the right cycle is contractible by Claim 2 (suitably relabeled), so $ABCX$ is contractible in that case too. We can therefore assume that none of the segments $[AC]$, $[BX]$, $[BY]$, or $[BZ]$ is a Rips edge.

Now let M be a common point of intersection of AB, CD, and XYZ. Lemma 6.2.4 implies that $|BM| > \frac{1}{2}$, and so $|AM| = |AB| - |BM| \leq \frac{1}{2}$. Since $|AC| > 1$, we have $|CM| = |AC| - |AM| > \frac{1}{2}$, and so $|DM| = |CD| - |CM| \leq \frac{1}{2}$. These inequalities imply that $|AD| \leq |AM| + |DM| \leq 1$, so $[AD]$ is a Rips edge.

It follows that $\mathcal{R}$ contains the cycle $ADCX$. This cycle is homotopic to $ABCX$, since $\langle ABCD \rangle$ is a cone by Proposition 6.2.1. Lemma 6.2.4 implies that at least one of the segments $[DX]$, $[DY]$, $[DZ]$ must be a Rips edge. Arguing as before, with D in place of B, we conclude that $ADCX$, and thus $ABCX$, is contractible. This completes the proof of Claim 3. □

6.2.3 Lifting Paths via Chaining

For any path α in $\mathcal{R}^1$, the projection $p(\alpha)$ is a path in $\mathcal{S}^1$, but not every shadow path is the projection of a Rips path. Every oriented shadow edge in $\mathcal{S}$ is covered by one or more oriented edges in $\mathcal{R}$. Thus to every path in $\mathcal{S}^1$ can be associated a sequence of oriented edges in $\mathcal{R}$. These edges do not necessarily form a path, but projections of consecutive Rips edges necessarily intersect at a shadow vertex.

Let $[AB]$ and $[CD]$ be oriented Rips edges induced by consecutive edges in some shadow path. A *chaining sequence* is a path from A to D in the subcomplex $\langle ABCD \rangle$ which begins with the edge AB and ends with the edge CD.

If we concatenate chaining sequences of shadow edges in $\mathcal{S}$ by identifying the Rips edges in the beginning and end of adjacent lifting sequences, we obtain a *lift* of the shadow path to $\mathcal{R}$. For any shadow path α in $\mathcal{S}$, we let $\widehat{\alpha}$ denote a lift of α to the Rips complex by means of chaining sequences. Note that the lift of a shadow path is not a true lift with respect to the projection map p — the endpoints, for example, may differ.

Lemma 6.2.6. *For any path α in $\mathcal{S}^1$, any two lifts of α to $\mathcal{R}$ with the same endpoints are homotopic in $\mathcal{R}$ rel endpoints.*

Proof: Let σ and τ be consecutive shadow edges in α, and let $[AB]$ and $[CD]$ be Rips edges such that $\sigma \subseteq AB$ and $\tau \subseteq CD$. Proposition 6.2.1 implies that all chaining sequences from A to D are homotopic rel endpoints in $\langle ABCD \rangle$, and thus in $\mathcal{R}$. If every shadow edge in α lifts to a unique Rips edge, the proof is complete.

On the other hand, suppose $\tau \subseteq CD \cap C'D'$ for some Rips edge $[C'D']$ that overlaps $[CD]$. Proposition 6.2.1 implies that both $[CC']$ and $[DD']$ are Rips

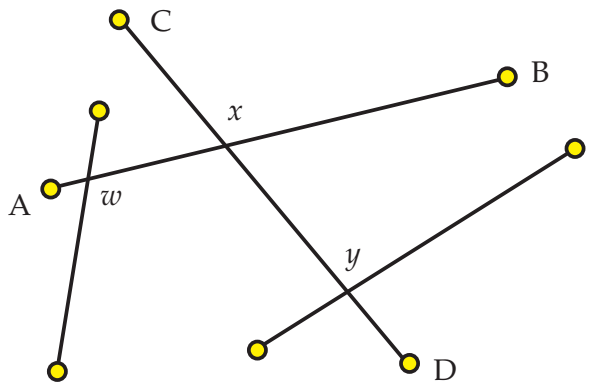

Figure 6.5. The setting for Lemma 6.2.7

edges. Moreover, since AB intersects $CD \cap C'D'$, any chaining sequence from A to D is homotopic rel endpoints in $\mathcal{R}$ to any chaining sequence from A to D' followed by $[D'D]$. Thus, concatenation of chaining sequences is not dependent on uniqueness of edge lifts. □

We next show that the projection of a lift of any two consecutive shadow edges is homotopic to the original edges.

Lemma 6.2.7. *For any two adjacent shadow edges wx and xy, where AB and CD are Rips edges with $wx \subseteq AB$ and $xy \subseteq CD$, $p(\widehat{wx \cdot xy})$ is homotopic rel endpoints to the path $ABxCD$ in $\mathcal{S}$.*

Proof: Consider the possible chaining sequences from A to D for $wx \cdot xy$. Either BC or AD must exist in $\mathcal{R}$ by Proposition 6.2.1.

Suppose BC exists. By Lemma 6.2.6, the chaining sequence is the Rips path $ABCD$ (up to homotopy rel endpoints). Either the triangle $[ABC]$ or the triangle $[BCD]$ exists in $\mathcal{R}$ by Proposition 6.2.1, so the triangle BCx is in shadow. This gives that $ABCD \simeq AxD \simeq ABxCD$ in $\mathcal{S}$.

If BC is not a Rips edge, then AD must be a Rips edges. By Lemma 6.2.6, the chaining sequence is the Rips path $ABADCD$ (up to homotopy rel endpoints). Either the triangle $[ACD]$ or the triangle $[ABD]$ exists in $\mathcal{R}$ by Proposition 6.2.1. Therefore, ADx lies in the shadow, so we get $ABADCD \simeq ABxCD$ in $\mathcal{S}$. □

Lemma 6.2.8. *For any lift $\widehat{\alpha}$ of any shadow path α with endpoints in $p(\mathcal{R}^{(0)})$, we have $p(\widehat{\alpha}) \simeq \alpha$ rel endpoints.*

Proof: For each pair of edges consecutive shadow edges wx and xy in α, where $wx \subseteq AB$, $xy \subseteq CD$, and AB and CD are Rips edges, Lemma 6.2.7 says that the projection of their lifting sequence deforms back to $ABxCD$. Every adjacent pair of chaining sequences can still be identified along common edges,

since each ends with the first edge in the next one along α. The projection is homotopic rel endpoints to the original path α except for spikes of the form xB and xC at each shadow junction, which can be deformation retracted, giving $p(\widehat{\alpha}) \simeq \alpha$. □

6.3 1-Connectivity on $\mathbb{R}^2$

The following is the main theorem of this paper.

Theorem 6.3.1. *For any set of points in $\mathbb{E}^2$, $\pi_1(p)\colon \pi_1(\mathcal{R}) \to \pi_1(\mathcal{S})$ is an isomorphism.*

Proof: Assume that all π_1 computations are performed with a basepoint in $p(\mathcal{R}^{(0)})$, to remove ambiguity of endpoints in lifts of shadow paths to $\mathcal{R}$. Surjectivity of p on $\pi_1(\mathcal{S})$ follows from Lemma 6.2.8 and the fact that any loop in $\mathcal{S}$ is homotopic to a loop of shadow edges.

To prove injectivity, note that any contractible cycle in $\mathcal{S}$ can be formed by concatenating boundary loops of shadow faces (conjugated to the basepoint). Using Lemma 6.2.8, injectivity of $\pi_1(p)$ will follow by showing that the boundary of any shadow face lifts to a contractible loop in $\mathcal{R}$. Consider therefore a shadow face Ψ contained in the projection of a Rips 2-simplex $[XYZ]$, and choose $[XYZ]$ to be minimal in the partial order of such 2-simplices generated by inclusion on the projections.

Write $\partial\Psi$ as $\alpha_1 \cdot \alpha_2 \cdots \alpha_n$, where the α_i are the shadow edges, and let $[A_iB_i]$ be a sequence of directed Rips edges with $\alpha_i \subseteq [A_iB_i]$. See Figure 6.6. Neither the A_i nor the B_i project to the interior of XYZ. If any Rips vertex W did so, the edges $[XW]$, $[YW]$ and $[ZW]$ would exist in $\mathcal{R}$. Since Ψ is a shadow face, it cannot be split by the image of any of these three edges. Therefore, Ψ must be contained in the projected image of a Rips 2-simplex, say $[XYW]$, whose image lies within that of $[XYZ]$, contradicting the minimality assumption on $[XYZ]$. The hypotheses of Proposition 6.2.3 thus apply to $[XYZ]$ and the consecutive edges $[A_iB_i]$, $[A_{i+1}B_{i+1}]$, giving that each complex $\langle A_iB_iA_{i+1}B_{i+1}XYZ\rangle$ is simply connected.

Fix the vertex X as a basepoint and fix a sequence of edge paths β_i in $\langle A_iB_iXYZ\rangle$ from X to A_i. Such paths exist and are unique up to homotopy since $\langle A_iB_iXYZ\rangle$ is a cone by Proposition 6.2.2. We decompose $\widehat{\partial\Psi}$ into loops $\gamma_1 \cdots \gamma_n$, where γ_i is the loop with basepoint X given by

$$\gamma_i = \beta_i \cdot (\widehat{\alpha_i \cdot \alpha_{i+1}}) \cdot [B_{i+1}A_{i+1}] \cdot \beta_{i+1}^{-1}$$

where all indices are computed modulo n. By Proposition 6.2.3, each of these loops γ_i is contractible; hence, so is $\widehat{\Psi}$. □

Corollary 6.3.2. *The fundamental group of a Rips complex of a planar point set is free.*

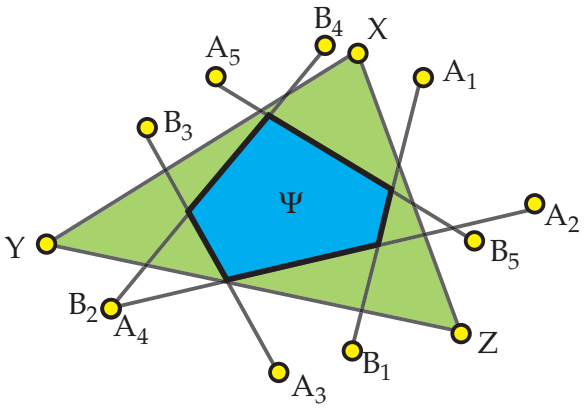

Figure 6.6. The boundary of a shadow face Ψ within XYZ is determined by Rips edges $[A_iB_i]$ whose projected endpoints lie outside XYZ.

Proof: $\mathcal{S}$ is a polygonal region (possibly with holes) in the plane, so $\pi_1(\mathcal{S})$ is a free group. Since $\pi_1(\mathcal{R}) \simeq \pi_1(\mathcal{S})$, the result follows. □

6.4 Quasi Rips Complexes and Shadows

We observe that Theorem 6.3.1 fails for quasi-Rips complexes, even for those with arbitrarily small uncertainty intervals. The failure of Proposition 6.2.1 in the quasi-Rips case makes it a simple exercise for the reader to generate examples of a quasi-Rips complexes which are simply-connected but whose shadows are not. Worse failure than this is possible.

Theorem 6.4.1. *Given any uncertainty interval (ϵ, ϵ') and any finitely presented group G, there exists a quasi-Rips complex $\mathcal{R}_Q$ with $\pi_1(\mathcal{R}_Q) \cong G * F$, where F is a free group.*

Proof: It is well-known that any finitely presented group G can be realized as the fundamental group of a 2-dimensional cell complex whose 1-skeleton is a wedge of circles over the generators and whose 2-cells correspond to relations. Such a complex can be triangulated, and, after a barycentric subdivision, can be assumed to be 3-colored: that is, there are no edges between vertices of the same color. Call this vertex 3-colored 2-d simplicial complex K.

We perform a 'blowup' of the complex K to a 3-d simplicial complex $\tilde{K}$ as follows (see Figure 6.7 for an example). Recall, the geometric realization of K can be expressed as the disjoint union of closed i-simplices with faces glued via simplicial gluing maps (the Δ-complex [73]). To form $\tilde{K}$, take the disjoint union of closed i-simplices of K and instead of simplicial gluing maps, use the **join** to connect all faces. The 3-coloring of K is inherited by $\tilde{K}$ via the blowup process.

There is a natural collapsing map $c : \tilde{K} \to K$ which collapses the joins to simplicial identification maps. The inverse image of any point in an open 2-simplex (1-simplex, resp.) of K is a closed 0-simplex (2-simplex resp.) of $\tilde{K}$.

The inverse image of a vertex $v \in K$ consists of the 1-skeleton of the link of v in K. If we fill in $\tilde{K}$ by taking the flag completion, then $c^{-1}(v)$ is a copy of the star of v in K. Thus, upon taking the flag complex of $\tilde{K}$, the fiber of c for each point in K is contractible, which shows that the flag complex of $\tilde{K}$ is homotopic to K and thus preserves π_1.

We now embed $\tilde{K}$ in a quasi-Rips complex $\mathcal{R}_Q$. Define the vertices of $\mathcal{R}_Q$ in $\mathbb{R}^2$ as follows. Fix an equilateral triangle of side length $(\epsilon + \epsilon')/2$ in $\mathbb{R}^2$. Embed the vertices of $\tilde{K}$ arbitrarily in sufficiently small open balls (of radii no larger than $(\epsilon' - \epsilon)/4$) centered at the vertices of this triangle, respecting the 3-coloring. For this vertex set in $\mathbb{R}^2$, we define $\mathcal{R}_Q$ by placing an edge between vertices according to the edges of $\tilde{K}$, using the fact that any two vertices not of the same color are separated by a distance within the uncertainty interval. Of course, we must also add a complete connected graph on all vertices with a given color, since these lie within the small balls.

The quasi-Rips complex $\mathcal{R}_Q$ is the flag complex of this graph. It contains the flag complex of $\tilde{K}$, along with three high-dimensional simplices, one for each color.

We claim that any 2-simplex of $\mathcal{R}_Q$ which is not also a 2-simplex of $\tilde{K}$ has all vertices of the same color. Proof: Consider a 2-simplex $\sigma \in \mathcal{R}_Q$ spanning more than one color. Since the only edges added to form $\mathcal{R}_Q$ from $\tilde{K}$ have both ends with identical colors, it must be that $\sigma \cap \tilde{K}$ contains two edges which share a vertex. Any two edges in $\tilde{K}$ which share a vertex are sent by the collapsing map c to either (1) two edges of a 2-simplex in K; or (2) a single 1-simplex of K; or (3) a single vertex of K. In either case, the entire 2-simplex σ exists in the flag complex of $\tilde{K}$.

We end by showing that $\pi_1(\mathcal{R}_Q)$ is a free extension of G. Each of the three large colored simplices added to form $\mathcal{R}_Q$ from $\tilde{K}$ is homotopy equivalent to adding an abstract colored vertex (the apex of the cone) and an edge from this apex to the blowup of each 0-simplex of K in $\tilde{K}$. This is homotopy equivalent to taking a wedge with (many) circles and thus yields a free extension of the fundamental group of the flag complex of $\tilde{K}$, G. □

We note that the construction above may be modified so that the lower-bound Rips complex $\mathcal{R}_\epsilon$ is connected. If necessary, the complex can be so constructed that the inclusion map $\mathcal{R}_\epsilon \hookrightarrow \mathcal{R}_{\epsilon'}$ induces an isomorphism on π_1 (which factors through $\pi_1(\mathcal{R}_Q)$).

6.5 k-Connectivity in $\mathbb{R}^n$

Theorem 6.3.1 points to the broader question of whether higher-order topological data are preserved by the shadow projection map. Recall that a topological space is **k-connected** if the homotopy groups π_i vanish for all $0 \leq i \leq k$. A map between topological spaces is k-connected if the induced homomorphisms

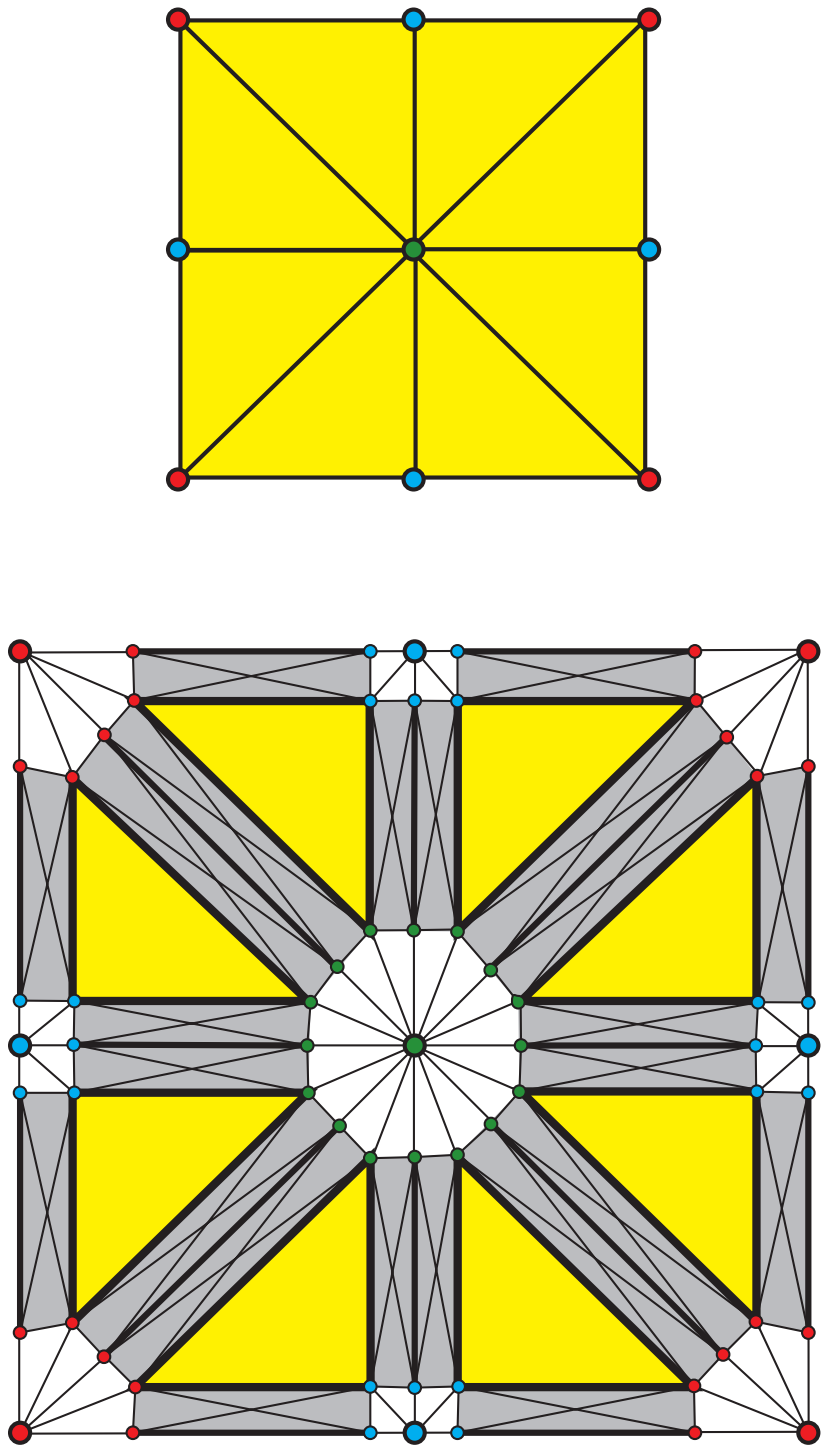

Figure 6.7. A 3-colored simplicial complex K and its blowup $\tilde{K}$, whose flag completion is homotopy equivalent to K. Opposite edges of K (and thus $\tilde{K}$) can be identified to yield a torus, projective plane, or Klein bottle.

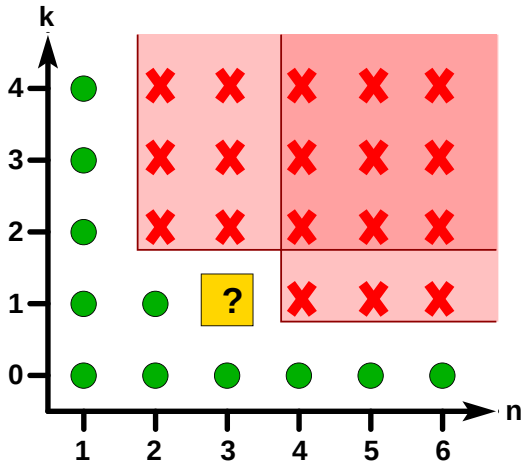

Figure 6.8. For which (n, k) is the Rips projection map in $\mathbb{E}^n$ k-connected? The only unresolved case is $(3, 2)$.

on π_i are isomorphisms for all $0 \leq i \leq k$.

We summarize the results of this section in Figure 6.8.

Throughout this paper, we have ignored basepoint considerations in the description and computation of π_1. The following proposition justifies this assumption.

Proposition 6.5.1. *For any set of points in $\mathbb{E}^n$, the map $p\colon \mathcal{R} \to \mathcal{S}$ is 0-connected.*

Proof: Certainly $\pi_0(p)$ is surjective, since p is surjective. The injectivity of $\pi_0(p)$ is a consequence of the following claim: If two Rips simplices σ and τ have intersecting shadows, then σ and τ belong to the same connected component of $\mathcal{R}$.

To prove the claim, suppose that $p(\sigma)$ and $p(\tau)$ intersect. By translation, we can suppose that $0 \in p(\sigma) \cap p(\tau)$. If $\{x_i\}$ and $\{y_j\}$ respectively denote the vertices of σ and τ, then

$$\sum_i \lambda_i x_i = 0 = \sum_j \mu_j y_j$$

for suitable convex coefficients $\{\lambda_i\}$ and $\{\mu_j\}$. Then

$$\begin{aligned}
\sum_{i,j} \lambda_i \mu_j |x_i - y_j|^2 &= \sum_{i,j} \lambda_i \mu_j |x_i|^2 - 2 \sum_{i,j} \lambda_i \mu_j (x_i \cdot y_j) + \sum_{i,j} \lambda_i \mu_j |y_j|^2 \\
&= \sum_i \lambda_i |x_i|^2 - 2 \sum_i \lambda_i x_i \cdot \sum_j \mu_j y_j + \sum_j \mu_j |y_j|^2 \\
&= \sum_i \lambda_i |x_i|^2 + \sum_j \mu_j |y_j|^2,
\end{aligned}$$

and similarly

$$\begin{aligned}\sum_{i,i'} \lambda_i \lambda_{i'} |x_i - x_{i'}|^2 &= 2 \sum_i \lambda_i |x_i|^2, \\ \sum_{j,j'} \mu_j \mu_{j'} |y_j - y_{j'}|^2 &= 2 \sum_j \mu_j |y_j|^2.\end{aligned}$$

Since every edge $x_i x_{i'}$ and $y_j y_{j'}$ has length at most 1, the left-hand sides of these last equations have value at most 1. Thus $\sum_i \lambda_i |x_i|^2 \leq 1/2$ and $\sum_j \mu_j |y_j|^2 \leq 1/2$. It follows that $\sum_{i,j} \lambda_i \mu_j |x_i - y_j|^2 \leq (1/2) + (1/2) = 1$ and so at least one edge $x_i y_j$ has length at most 1.

Thus the simplices σ, τ are connected by an edge, as required. □

Proposition 6.5.2. *For any set of points in $\mathbb{E}^1$, the map $p\colon \mathcal{R} \to \mathcal{S}$ is a homotopy equivalence.*

Proof: Both $\mathcal{R}$ and $\mathcal{S}$ are homotopy equivalent to finite unions of closed intervals in $\mathbb{E}^1$, and therefore to finite sets of points. This is clear for $\mathcal{S}$. For $\mathcal{R}$, we note that $\mathcal{R}_1$ is equal to the Čech complex $\mathcal{C}_1$ in $\mathbb{E}^1$. Certainly the two complexes have the same 1-skeleton. Moreover, Helly's theorem implies that Čech complexes are flag complexes in 1D: a collection of convex balls has nonempty intersection if all pairwise intersections are nonempty. Thus $\mathcal{R}_1 = \mathcal{C}_1$. By the nerve theorem, this complex has the homotopy type of a union of closed intervals in $\mathbb{E}^1$.

Since a 0-connected map between finite point sets is a homotopy equivalence, the same conclusion now holds for the 0-connected map $p\colon \mathcal{R} \to \mathcal{S}$. □

Proposition 6.5.3. *There exists a configuration of points in $\mathbb{E}^2$ for which p is not 2-connected.*

Proof: Consider the vertices $rx_1, rx_2, rx_3, rx_4, rx_5, rx_6$ of a regular hexagon of radius $s$centered at the origin. If $1/2 < r \leq 1/\sqrt{3}$ then only the three main diagonals are missing from $\mathcal{R}$. Thus $\mathcal{R}$ has the structure of a regular octahedron, and therefore the homotopy type of a 2-sphere. On the other hand $\mathcal{S}$ is just the hexagon itself (including interior), and is contractible. (See Figure 6.9 □

The example of Proposition 6.5.3 extends to higher homotopy groups by constructing cross-polytopes, as in [38].

Proposition 6.5.4. *There exists a configuration of points in $\mathbb{E}^4$ for which p is not 1-connected.*

Proof: Consider the six points

$$(rx_1, \epsilon x_1), \quad (rx_2, 0), \quad (rx_3, \epsilon x_3), \quad (rx_4, 0), \quad (rx_5, \epsilon x_5), \quad (rx_6, 0)$$

in $\mathbb{E}^4$, in the notation of the previous proposition. Then $\mathcal{R}$ has the structure of a regular octahedron, but the map $p\colon \mathcal{R} \to \mathcal{S}$ identifies one pair of antipodal

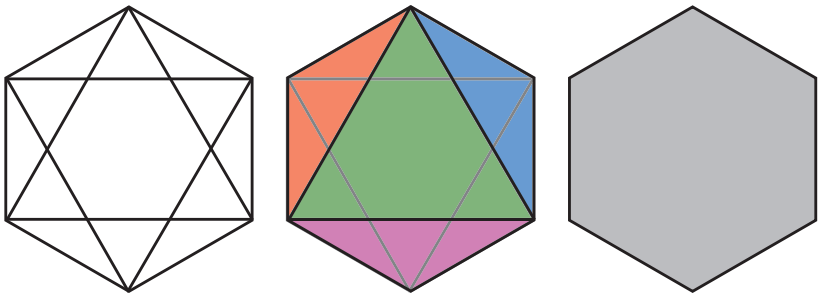

Figure 6.9. Left: P is 6 points in $\mathbb{R}^2$ placed evenly around a disk of radius $5/4$. Middle: $\mathcal{R}(P)$ will be an octahedron (and thus homeomorphic to S^2). Right: $\mathcal{S}(P)$ will be a topological disk.

points (specifically, the centers of the two large triangles, 135 and 246). Thus $\mathcal{R}$ is simply-connected, whereas $\pi_1(\mathcal{S}) = \mathbb{Z}$. □

We note that these counterexamples may be embedded in higher dimensions and perturbed to lie in general position.

6.6 Algorithmic Results

In this section, we briefly discuss recent algorithmic results for planar Rips complexes [27]. Note that this work appeared in the Masters thesis of Pratik Worah, and is included in this thesis only for the sake of completeness.

6.6.1 Structural Results

In [27], we develop an efficient algorithm to compute the Rips shadow $\mathcal{S}(P)$ of a given set P of n points in the plane. Our algorithm relies on two structural results, which may be of independent interest. First, although the Rips complex $\mathcal{R}(P)$ can have $\Theta(n^2)$ edges and $\Theta(n^3)$ triangles in the worst case, the Rips shadow $\mathcal{S}(P)$, which is the union of those edges and triangles, always has complexity $O(n)$. Second, there is a subset of $O(n)$ Rips edges and Rips triangles whose union is the entire the Rips shadow $\mathcal{S}(P)$.

Theorem 6.6.1. *The Rips shadow of n points in the plane has combinatorial complexity $O(n)$.*

Proof: Fix a set P of n points in the plane. We assume without loss of generality that $\mathcal{R}(P)$ and therefore $\mathcal{S}(P)$ are connected; if not, we can analyze each connected component independently. This assumption implies that each hole in $\mathcal{S}(P)$ has a single boundary cycle.

We bound the complexity of the Rips shadow by (over-)counting the number of boundary edges and vertices. The same boundary vertex or edge may appear multiple times on the same facial walk or on multiple walks; we count each occurrence separately. To simplify our presentation, we consider the two sides of any Rips edge or shadow boundary edge separately; for any edge uv, let $\overrightarrow{uv}$ and

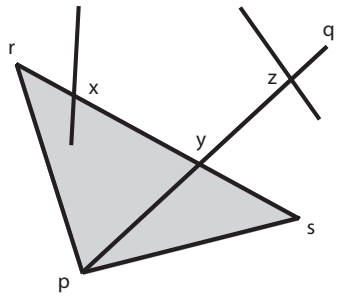

Figure 6.10. If the Rips triangle prs is present, then py is in shadow, and y is the boundary corner on pq which is closest to p.

$\overrightarrow{vu}$ denote its two oriented *halfedges*. A facial walk now consists of a sequence of boundary halfedges, oriented with the hole on the left; two consecutive boundary halfedges $\overrightarrow{xy}$ and $\overrightarrow{yz}$ form a *boundary corner* at y. We prove that there are $O(n)$ boundary corners.

We say that a Rips halfedge $\overrightarrow{pq}$ is *uncovered* if there is no Rips triangle pqr with r to the left of the oriented line $\overrightarrow{pq}$. Every corner of the shadow boundary is located at the intersection of two uncovered halfedges, possibly at a common endpoint.

If $\overrightarrow{pq}$ and $\overrightarrow{pr}$ are two uncovered Rips halfedges with a common source point p, then $\angle qpr > \pi/3$ since qr cannot be a Rips edge. It follows that any point in P is the source of at most five uncovered halfedges, giving at most $5n$ uncovered edges total. In addition, there are at most five boundary corners at any point in P.

Let $\overrightarrow{pq}$ and $\overrightarrow{rs}$ be uncovered Rips halfedges, with r to the left of $\overrightarrow{pq}$, whose interiors intersect at boundary vertex y. Suppose some pair of boundary halfedges $\overrightarrow{xy} \subset \overrightarrow{pq}$ and $\overrightarrow{yz} \subset \overrightarrow{rs}$ form a boundary corner at y. Either prs or pqs is a Rips triangle, since either of the other two possible triangles would cover $\overrightarrow{pq}$ or $\overrightarrow{rs}$. If prs is a Rips triangle, segment py lies inside the shadow, so y is the closest boundary corner to p, among all boundary corners on $\overrightarrow{pq}$. See Figure 6.10. Similarly, if pqs is a Rips triangle, y is the boundary corner on $\overrightarrow{rs}$ that is closest to s.

Thus, every boundary corner that is not a point in P is either the first or last boundary corner on some uncovered halfedge. It follows that there are at most $10n$ boundary corners not at points in P, and thus at most $15n$ boundary corners overall. □

Theorem 6.6.2. *For any set P of n points in the plane, there is a set of $O(n)$ Rips edges and Rips triangles whose union is the Rips shadow $\mathcal{S}(P)$.*

The proof of Theorem 6.6.2 takes an existing Rips complex $\mathcal{R}'$ and adds another Rips vertex to it, getting a larger complex $\mathcal{R}$. It then shows that the regions in $\mathcal{S} \setminus \mathcal{S}'$ can be covered using at most 24 new edges or triangles. We then get (by induction) that a Rips complex with n points can be covered using at most $24n$ triangles and edges.

Theorem 6.6.3 (No small holes). *Any hole in the Rips shadow of a set of points in the plane has circumradius at least* $(\sqrt{2}-1)/8\sqrt{3} \approx 0.029893$.

The proof of Theorem 6.6.3 takes any hole in $\mathcal{S}$ and uses case analysis combined with the triangle inequality and Prop 6.2.1 to force the hole to be large.

6.6.2 Algorithmic Results

We next outline the algorithms developed for computing $\mathcal{S}$ and testing homotopy properties in $\mathcal{R}$.

Theorem 6.6.4. *Given a set P of n points in the plane, we can construct $\mathcal{S}(P)$ in $O((m+n)\log n)$ time, where m is the number of edges in the proximity graph of P.*

We briefly sketch the algorithm for completeness. To compute $\mathcal{S}$, we impose a grid of cells with size $1/2 \times 1/2$ over the point set. There are $O(n)$ grid cells which contain points, and for any particular Rips vertex p, the set of cells which contain Rips vertices connected to p is of size $O(1)$. We can then find, for each Rips vertex, the 24 edges and triangles mentioned in Theorem 6.6.2 using fairly standard algorithms from computational geometry. See [27] for more details.

Theorem 6.6.5. *After $O(m \log n)$ preprocessing time, we can determine whether any given cycle of k Rips edges is contractible in $\mathcal{R}(P)$, either in $O(k \log n)$ time using $O(n)$ space, or in $O(k)$ time using $O(m)$ space.*

Our algorithm follows a standard strategy used by Cabello *et al.* [18] and many other authors [77, 45, 42, 33, 25, 34, 35] for encoding the homotopy class of paths and cycles in two-dimensional spaces. We compute a sequence of line segments $\phi_1, \phi_2, \ldots, \phi_b$, which we call *fences*, that form a spanning tree of the holes; we assign each fence an arbitrary orientation. The *crossing word* of any cycle γ records the sequence of fences that γ crosses, along with the direction of each crossing. For example, the crossing word $12\overline{2}3$ indicates that γ first crosses ϕ_1 from left to right, then ϕ_2 from left to right, then ϕ_2 from right to left, and finally ϕ_3 from left to right. We can *reduce* any crossing word by removing any matching pairs of the form $x\overline{x}$ or $\overline{x}x$; each reduction corresponds to a continuous deformation of γ that removes some fence crossings. Finally, γ is contractible if and only if its reduced crossing word is empty. Our spanning tree construction is a straightforward consequence of Theorem 6.6.3.

Finally, we describe how to find the shortest cycle in the Rips complex that is noncontractible. We assume that each edge pq in the proximity graph has a non-negative weight $w(pq)$; the length of a cycle is the sum of the weights of its edges. Our results hold for *any* non-negative edge weights; in particular, we can minimize either the number of edges or the total Euclidean length of the cycle.

For any point p and any Rips edge qr, let $C(p, qr)$ denote the cycle of Rips edges composed of the shortest path from p to q, the edge qr, and the shortest

path from $s$back to p. The following characterization of shortest noncontractible cycles was first observed by Thomassen [117, 95] for graphs embedded on surfaces; see also [58].

Lemma 6.6.6. *For any point $p \in P$, the shortest noncontractible cycle in $\mathcal{R}(P)$ that passes through p is the cycle $C(p, qr)$ for some Rips edge qr.*

Using the previous lemma, we get the following result.

Theorem 6.6.7. *Given a set P of n points in the plane, we can compute the shortest noncontractible cycle in $\mathcal{R}(P)$ in $O(n^2 \log n + mn)$ expected time.*

6.7 Conclusion

The relationship between a Rips complex and its projected shadow is extremely delicate, as evidenced by the universality result for quasi-Rips complexes (Theorem 6.4.1) and the lack of general k-connectivity in $\mathbb{R}^n$ (§6.5). These results act as a foil to Theorem 6.3.1: it is by no means *a priori* evident that a planar Rips complex should so faithfully capture its shadow.

We close with a few remarks and open questions.

1. Are the cross-polytopes of Proposition 6.5.3 the only significant examples of higher homology in a (planar) Rips complex? If all generators of the homology $H_k(\mathcal{R})$ for $k > 1$ could be classified into a few such 'local' types, then, after a local surgery on $\mathcal{R}$ to eliminate higher homology, one could use the Euler characteristic combined with Theorem 6.3.1 as a means of quickly computing the number of holes in the shadow of a planar Rips complex. This method would have the advantage of being local and thus distributable.

2. Does the projection map preserve π_1 for a Rips complex of points in $\mathbb{R}^3$? Our proofs for the 2-d case rest on some technical lemmas whose extensions to 3-d would be neither easy to write nor enjoyable to read. A more principled approach would be desirable, but is perhaps not likely given the 1-connectivity on $\mathbb{R}^3$ is a borderline case.

3. The algorithmic results described from [27] rely heavily on knowing the exact coordinates of the points. However, the primary advantage of Rips complexes over other alternatives is the fact that we can avoid exact distance computations by simply knowing connectivity information. Is there any way to compute the shadow or homotopy properties of given cycles without exact coordinates?

References

[1] Stephen Alstrup, Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Transactions on Algorithms*, 1(2):243–264, 2005.

[2] Helmut Alt and Maike Buchin. Semi-computability of the Fréchet distance between surfaces. In *Proceedings of the 21st European Workshop on Computational Geometry*, pages 45–48, March 2005.

[3] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5(1–2):75–91, 1995.

[4] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the 14th annual Symposium on Computational Geometry*, pages 39–48. ACM Press, 1998.

[5] Reinhold Baer. Isotopie von kurven auf orientierbaren geschlossenen flächen und ihr zusammenhang mit der topologischen deformation der flächen. *J. f. Math.*, 159:101–116, 1928.

[6] Lali Barrière, Pierre Fraigniaud, and Lata Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2001.

[7] Saugata Basu. Polynomial time algorithm for computing the top betti numbers of semi-algebraic sets defined by quadratic inequalities. In *Proceedings of the 37th annual ACM Symposium on Theory of Computing*, pages 313–322, 2005.

[8] Saugata Basu. Computing the first few betti numbers of semi-algebraic sets in single exponential time. *Journal of Symbolic Computation*, 41(10):1125–1154, 2006.

[9] Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. In *Proceedings of the 14th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 609–617, 2003.

[10] Sergei Bespamyatnikh. Encoding homotopy of paths in the plane. In *Proceedings of LATIN 2004: Theoretical Infomatics*, volume 2976 of *Lect. Notes Comput. Sci.*, pages 329–338. Springer-Verlag, 2004.

[11] Anders Björner. *Topological methods*, volume 2, pages 1819–1872. 1995.

[12] Johannes Blömer. Computing sums of radicals in polynomial time. In *Proceedings of the 32nd annual IEEE Symposium on Foundations of Computer Science*, pages 670–677, 1991.

[13] William W. Boone. The word problem. *Proceedings of the National Academy of Sciences 44*, 10:1061–1065, 1958.

[14] Martin R. Bridson and Andr Haefliger. *Metric Spackes of Non-Positive Curvature*. A Series of Comprehensive Studies in Mathematics. Springer-Verlag, 1999.

[15] Sergio Cabello. Many distances in planar graphs. In *Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1213–1220, 2006.

[16] Sergio Cabello and Erin W. Chambers. Multiple source shortest paths in a genus g graph. In *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 89–97, 2007.

[17] Sergio Cabello, M. DeVos, Jeff Erickson, and Bojan Mohar. Finding one tight cycle. In *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms*, page to appear, 2008.

[18] Sergio Cabello, Yuanxin Liu, Andrea Mantler, and Jack Snoeyink. Testing homotopy for paths in the plane. *Discrete and Computational Geometry*, 31(1):61–81, 2004.

[19] Sergio Cabello and Bojan Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discrete and Computational Geometry*, 37(2):213–235, 2007.

[20] E. Carlsson, Gunnar Carlsson, and Vin de Silva. An algebraic topological method for feature identification. *International Journal of Computational Geometry and Applications*, 16:291–314, 2006.

[21] Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. preprint, 2006.

[22] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11:149–187, 2005.

[23] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In *Fifth International Conference on Computer Vision*, pages 694–699, 1995.

[24] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite. Walking your dog in the woods in polynomial time. In *Proceedings of the 24th annual ACM Symposium on Computational Geometry*, 2008.

[25] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. In *Proceedings of the 22nd annual Symposium on Computational geometry*, pages 421–429, New York, NY, USA, 2006. ACM Press.

[26] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. In *Proceedings of the 22nd annual Symposium on Computational Geometry*, pages 421–429. ACM, 2006.

[27] Erin W. Chambers, Jeff Erickson, and Pratik Worah. Testing contractibility in planar rips complexes. In *Proceedings of the 24th annual Symposium on Computational Geometry*, 2008.

[28] Bernard Chazelle. A theorem on polygon cutting with applications. In *Proceedings of the 23rd annual IEEE Symposium on Foundations of Computer Science*, pages 339–349, 1982.

[29] Jindong Chen and Yijie Han. Shortest paths on a polyhedron, part I: Computing shortest paths. *International Journal on Computational Geometry and Applications*, 6(2):127–144, 1996.

[30] David L. Chopp and James A. Sethian. Flow under curvature: Singularity formation, minimal surfaces, and geodesics. *Journal of Experimental Mathematics*, 2(4):235–255, 1993.

[31] Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, January 1987.

[32] Richard Cole. Parallel merge sort. *SIAM Journal on Computing*, 17(4):770–785, 1988.

[33] Éric Colin de Verdière and Jeff Erickson. Tightening non-simple paths and cycles on surfaces. In *Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithm*, pages 192–201, 2006.

[34] Éric Colin de Verdière and Francis Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. In *Proceedings of the 11th Symposium on Graph Drawing*, volume 2912 of *Lecture Notes Comput. Sci.*, pages 478–490. Springer-Verlag, 2003.

[35] Éric Colin de Verdière and Francis Lazarus. Optimal systems of loops on an orientable surface. *Discrete and Computational Geometry*, 33(3):507–534, 2005.

[36] Altas Cook and Carola Wenk. Geodesic Fréchet and Hausdorff distance inside a simple polygon. Tech. Rep. CS-TR-2007-004, University of Texas at San Antonio, 2007.

[37] Vin de Silva and Gunnar Carlsson. Topological estimation using witness complexes. In *Symposium on Point-Based Graphics*, pages 157–166, 2004.

[38] Vin de Silva and Robert Ghrist. Coordinate-free coverage in sensor networks with controlled boundaries via homology. *International Journal of Robotics Research*, 25:1205–1222, 2006.

[39] Vin de Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic and Geometric Topology*, 7:339–358, 2007.

[40] Max Dehn. ber unendliche diskontinuierliche gruppen. *Mathematische Annalen*, 71(1):116–144, 1911.

[41] Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Bojan Mohar. Approximation algorithms via contraction decomposition. In *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 278–287, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[42] Tamal K. Dey and Sumanta Guha. Transforming curves on surfaces. *J. Comput. Sys. Sci.*, 58(2):297–325, 1999.

[43] Tamal K. Dey, Kuiyu Li, and Jian Sun. On computing handle and tunnel loops. In *Proceedings of the 2007 International Conference on Cyberworlds*, pages 357–366. IEEE Computer Society, 2007.

[44] Tamal K. Dey, Kuiyu Li, Jian Sun, and David Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3d models. In *Proceedings of the 35th annual Conference on Computer Graphics and Interactive Techniques*, 2008.

[45] Tamal K. Dey and Haijo Schipper. A new technique to compute polygonal schema for 2-manifolds with applications to null-homotopy detection. *Discrete and Computational Geometry*, 14:93–110, 1995.

[46] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[47] James R. Driscoll, Neil Sarnak, Daniel D. Sleator, and Robert E. Tarjan. Making data structures persistent. In *Proceedings of the 18th annual ACM Symposium on the Theory of Computing*, pages 109–121, New York, NY, USA, 1986. ACM Press.

[48] Jrgen Eckhoff. Helly, Radon, and Carathéodory type theorems. In *Handbook of convex geometry*, volume A, pages 389–448. 1993.

[49] Herbert Edelsbrunner and John Harer. Persistent homology: A survey. In J. Pack J. E. Goodman and R. Pollack, editors, *Discrete & Computational Topology: Twenty Years Later*. AMS Press, to appear.

[50] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28:511–533, 2002.

[51] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[52] Herbert Edelsbrunner and Afra Zomorodian. Computing linking numbers in a filtration. *Homology, Homotopy and Applications*, 5(2):19–37, 2003.

[53] Alon Efrat, Leonidas J. Guibas, Sariel Har-Peled, Joseph S. B. Mitchell, and T. M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete and Computational Geometry*, 28:535–569, 2002.

[54] Alon Efrat, Sariel Har-Peled, Leonidas J. Guibas, Joseph S.B. Mitchell, and T.M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete and Computational Geometry*, 28:535–569, 2002.

[55] Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry: Theory and Applications*, 35(3):162–172, 2006.

[56] David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[57] Jeff Erickson. Shortest paths on pl surfaces. Blog post, 2006.

[58] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete and Computational Geometry*, 31(1):37–59, 2004.

[59] Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1038–1046, 2005.

[60] Qing Fang, Jie Gao, and Leonidas J. Guibas. Locating and bypassing holes in sensor networks. *Mobile Networks and Applications*, 11(2):187–200, 2006.

[61] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987.

[62] Michael L. Fredman and Robert E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[63] Joel Friedman. Computing betti numbers via combinatorial laplacians. *Algorithmica*, 21:331–346, 1998.

[64] Robert Ghrist. Barcodes: The persistent topology of data. Preprint, 2007.

[65] Robert Ghrist and Abubakr Muhammad. Coverage and hole-detection in sensor networks via homology. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, page 34. IEEE Press, 2005.

[66] Dima Grigoriev and Anatol Slissenko. Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 17–24, 1998.

[67] Misha Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Mathematical Sciences Research Institute Publications*, pages 75–263. 1987.

[68] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of SIGGRAPH 2002*, pages 355–361, 2002.

[69] Leonidas Guibas. Kinetic data structures: A state of the art report, 1998.

[70] Igor Guskov and Zoë J. Wood. Topological noise removal. In *Proceedings of Graphics Interface 2001*, pages 19–26, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society.

[71] Wolfgang Haken. Theorie der normalflchen. *Acta Mathematica*, 105:245–375, 1961.

[72] Joel Hass. Algorithms for recognizing knots and 3-manifolds, 1997.

[73] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[74] Poul Heegaard and Max Dehn. Analysis situs. *Enzyklopdie der Mathematischen Wissenschaften*, 1907.

[75] Monika R. Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM*, 46(4):502–516, 1999.

[76] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class (extended abstract). In *Workshop on Algorithms and Data Structures*, pages 331–342, 1991.

[77] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4:63–67, 1994.

[78] John Hershberger and Subhash Suri. An optimal algorithm for euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

[79] Piotr Indyk and Anastasios Sidiropoulos. Probabilistic embeddings of bounded genus graphs into planar graphs. In *Proceedings of the 23rd annual Symposium on Computational Geometry*, pages 204–209, New York, NY, USA, 2007. ACM.

[80] Alon Itai, Christos H. Papadimitriou, and Jayme L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11:676–686, 1982.

[81] Ken-ichi Kawarabayashi and Bruce Reed. Computing crossing number in linear time. In *Proceedings of the 39th annual ACM Symposium on Theory of Computing*, pages 382–390, New York, NY, USA, 2007. ACM.

[82] Philip N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 146–155. SIAM, 2005.

[83] Donald E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1997.

[84] Fabian Kuhn and Aaron Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proceedings of the 2003 joint workshop on Foundations of mobile computing*, pages 69–78, 2003.

[85] Kazimierz Kuratowski. Sur le problme des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.

[86] Martin Kutz. Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time. In *Proceedings of the 22nd annual Symposium on Computational Geometry*, pages 430–438, 2006.

[87] D. T. Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.

[88] Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal Applied Mathematics*, 36:177–189, 1979.

[89] Anil Maheshwari and Jiehua Yi. On computing Fréchet distance of two paths on a convex polyhedron. In *Proceedings of the 21st European Workshop on Computational Geometry*, pages 41–44, 2005.

[90] A. A. Markov. Insolubility of the problem of homeomorphy. In *Proceedings of the International Congress of Mathematics*, pages 14–21. Cambridge University Press, 1958.

[91] Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30:852–866, 1983.

[92] Gary L. Miller and John H. Reif. Parallel tree contraction and its applications. In *Proceedings of the 26th annual IEEE Symposium on Foundations of Computer Science*, pages 478–489, 1985.

[93] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.

[94] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12:6–26, 1999.

[95] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, Baltimore, 2001.

[96] Abubakr Muhammad and Ali Jadbabaie. Dynamic coverage verification in mobile sensor networks via switched higher order laplacians. In *Robotics: Science and Systems*, 2007.

[97] Ketan Mulmuley, U.V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–114, 1987.

[98] Pyotr S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Proceedings of the Steklov Institute of Mathematics*, 44:1–143, 1955.

[99] Colm Ó Dúnlaing, Colum Watt, and David Wilkins. Homeomorphism of 2-complexes is equivalent to graph isomorphism. *International Journal of Computational Geometry and Applications*, 10(5):453–476, 2000.

[100] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[101] Michel Pocchiola and Gert Vegter. Computing the visibility graph via pseudo-triangulations. In *Symposium on Computational Geometry*, pages 248–257, 1995.

[102] Michel Pocchiola and Gert Vegter. Minimal tangent visibility graphs. *Computational Geometry: Theory and Applications*, 6, 1996.

[103] Neil Robertson and Paul D. Seymour. Graph minors i: Excluding a forest. *Journal of Combinatorial Theory Series B*, 35(1):39–61, 1983.

[104] Neil Robertson and Paul D. Seymour. Graph minors xi: Circuits on a surface. *Journal of Combinatorial Theory Series B*, 60(1):72–106, 1994.

[105] Neil Robertson and Paul D. Seymour. Graph minors xx: Wagner's conjecture. *Journal of Combinatorial Theory Series B*, 92(2):325–357, 2004.

[106] Jeanette P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM Journal on Computing*, 27(4):972–992, 1998.

[107] Raimund Seidel. The nature and meaning of perturbations in geometric computing. *Discrete and Computational Geometry*, 19:1–17, 1998.

[108] Alla Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Proceedings of Shape Modeling International*, 2002.

[109] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the 12th annual IEEE Visualization Conference*, pages 355–362, 2002.

[110] E. Sriraghavendra, K. Karthik, and C. Bhattacharyya. Frchet distance based approach for searching online handwritten documents. In *ICDAR07*, pages 461–465, 2007.

[111] Dvir Steiner and Anath Fischer. Cutting 3d freeform objects with genus-n into single boundary surfaces using topological graphs. In *Proceedings of the 7th annual ACM Symposium on Solid Modeling Applications*, pages 336–343, 2002.

[112] John Stillwell. *Classical Topology and Combinatorial Group Theory*. Springer-Verlag, New York, 1993.

[113] Jun-ya Takahashi, Hitoshi Suzuki, and Takao Nishizeki. Shortest noncrossing paths in plane graphs. *Algorithmica*, 16:339–357, 1996.

[114] Robert E. Tarjan. *Data Structures and Network Algorithms*. SIAM, Philadelphia, 1983.

[115] Robert E. Tarjan. Dynamic trees via Euler tours, applied to the network simplex algorithm. *Mathematical Programming: Series A and B*, 78:169–177, 1997.

[116] Robert E. Tarjan and Renato F. Werneck. Self-adjusting top trees. In *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 813–822. Society for Industrial and Applied Mathematics, 2005.

[117] Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *J. Comb. Theory Ser. B*, 48(2):155–177, 1990.

[118] Carsten Thomassen. Five-coloring maps on surfaces. *Journal of Combinatorial Theory Series B*, 59(1):89–105, 1993.

[119] René van Oostrum and Remco C. Veltkamp. Parametric search made practical. *Computational Geometry: Theory and Applications*, 28:75–88, 2004.

[120] Leopold Vietoris. über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97:454–472, 1927.

[121] E. F. Whittlesey. Classification of finite 2-complexes. *Proceedings of the American Mathematical Society*, 9:841–845, 1958.

[122] E. F. Whittlesey. Finite surfaces: a study of finite 2-complexes. i. local structure. *Mathematics Magazine*, 34:11–22, 1960.

[123] E. F. Whittlesey. Finite surfaces: a study of finite 2-complexes. ii. the canonical form. *Mathematics Magazine*, 34:67–80, 1960.

[124] Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, 2004.

[125] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, 2005.

Author's Biography

Erin Moriarty Wolf Chambers was born in Maryville, Illinois on March 27, 1980. She obtained her bachelor's degree in Computer Science from the University of Illinois at Urbana-Champaign in May 2002, and her Masters in Mathematics from the same institution in May 2006. After completing her PhD, Erin will begin working as a faculty member at Saint Louis University in Saint Louis, Missouri.